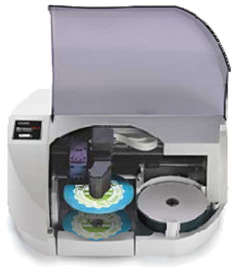
PTRobot API

DLL-Based Application Programming Interface



Primera Technology Inc.
March 1st, 2017 / Revision 2.6

1 PTRobot Description

1.1 Description

PTRobot is an API that allows developers to add robotic support for Primera duplicators to their own Windows applications. PTRobot consists of several Dynamic Link Libraries (DLLs) that application developers can utilize to move discs automatically (e.g. move a disc from an input bin into the recorder, etc.) and also provides the capability to print on the CD/DVD through the Surething CD Labeler application. PTRobot provides an easy method to create automated CD/DVD printing and/or recording applications (recording capability is not provided in PTRobot – for recording capability developers should use the PTBurn SDK from Primera). Currently, PTRobot provides support for the Disc Publisher II, Disc Publisher PRO, Disc Publisher XR, Disc Publisher XRP, and Disc Publisher SE, Disc Publisher SE-3, Disc Publisher Pro Xi-Series, Disc Publisher 4100 Series and Disc Publisher 4200 Series.

IMPORTANT NOTE: The PTRobot API functions use the WINAPI (which is _stdcall) calling convention.

1.2 Usage

Below is PSUEDO-CODE example for how a calling application should use PTRobot to implement robotics into its application.

At program startup:

```
PTRobot_Initialize(..)

PTRobot_EnumRobots(..)
- if the number of robots is greater than 1 then the calling application will need to
provide some logic/ui to determine which robot to use.  The app can use
PTRobot_GetRobotInfo(..) to get details about each robot.

PTRobot_EnumDrives(..) or PTRobot_EnumDrivesWithList(..)
- This will cause PTRobot to determine which drives are robotically controlled by
enumerating the drives themselves (PTRobot_EnumDrives(..)) or based off a list of drives
passed in (PTRobot_EnumDrivesWithList(..))

PTRobot_GetDriveInfo(..) for all drives returned
- This will allow the calling app to know which drive is which
```

Example a typical job:

```
PTRobot_LoadDrive(Robot, Drive, TRUE)
- This will load a disc into the drive from the input bin
(should set parameter 3 to TRUE if first round of the job)

PTRobot_GetRobotStatus(..)
- This would be called in a loop until a system error occurred or the system is idle.
 (NOTE: do not call in too tight of a loop – e.g. every 500ms or 1 second).

…Client application will now perform operations on the disc in the drive (e.g. record on
the disc).

      if the operations are successful
            PTRobot_LoadPrinterFromDrive(Robo, Drive)
            PTRobot_GetRobotStatus(..) called in a loop
            PTRobot_PrintFile(szPrintFile)  or  PTRobot_PrintFileWithMerge(…)
            PTRobot_GetRobotStatus(..) called in a loop
            PTRobot_UnloadPrinter(Robot, 0)
            PTRobot_GetRobotStatus(..) called in a loop
      else
            PTRobot_UnloadDrive(Robot, Drive, 100)
            PTRobot_GetRobotStatus(..) called in a loop
```

Before program exit:

PTRobot_Destroy()

2 API Functions

2.1 PTRobot Setup Functions

2.1.1 PTRobot_Initialize

```
///////////////////////////
//
//    PTRobot_Initialize
//
//    Description:
//          Function to initialize internal data structures of
//          the PTRobot module.
//    Params:
//          None
//    Notes:
//    Return:
//          PTROBOT_OK if Successful
//          PTROBOT_INTERNAL if an internal error occurred.
//
///////////////////////////
DWORD WINAPI PTRobot_Initialize();
```

2.1.2 PTRobot_Destroy

```
///////////////////////////
//
//    PTRobot_Destroy
//
//    Description:
//          Function to destroy internal data structures of
//          the PTRobot module.
//    Params:
//          None
//    Notes:
//    Return:
//          PTROBOT_OK if Successful
//          PTROBOT_SEQUENCE if this command is called out of sequence
//          PTROBOT_INTERNAL if an internal error occurred
//
///////////////////////////
DWORD WINAPI PTRobot_Destroy();
```

2.1.3 PTRobot_SetupDebugging

```
///////////////////////////
//
//    PTRobot_SetupDebugging
//
//    Description:
```

```
//          Function to setup logging in the PTRobot module.  We advise that your
//          application has a "back door" method of turning debugging on.  All debugging
//          is off by default.
//    Params:
//          szDbgFile         full path to a debug file
//          dwDbgLvl          debug Level (0-5)
//                                     0 = off, 1 = errors, 2 = warnings
//                                     3 = Info, 4 and 5 = more info
//          szTraceFile       full path to a trace file
//    Notes:
//          If szDbgFile is NULL then debugging will be turned off.  If szTraceFile is
//          NULL then function tracing will be off.  Function tracing just logs the
//          API function calls including the parameters.
//    Return:
//          PTROBOT_OK if Successful
//          PTROBOT_INTERNAL if an internal error occurred
//
///////////////////////////
DWORD WINAPI PTRobot_SetupDebugging(TCHAR * szDbgFile, DWORD dwDbgLvl, TCHAR *
szTraceFile);
```

2.1.4 PTRobot_EnumRobots

```
///////////////////////////
//
//    PTRobot_EnumRobots
//
//    Description:
//          Function to enumerate the Robots on the system.
//    Params:
//          phRobots    points to an array of HANDLEs to store
//                      the Robots found.
//          pdwNumRobots      points to a DWORD containing the number of HANDLEs
//                            in the phRobots array.  This value is an input
//                            and an output.  The user should specify the size
//                            (number of HANDLEs) of the phRobots array on input.
//                            The value of the pdwNumRobots on output will be the
//                             number of robots found.
//
//    Notes:
//      Both params will be updated upon successful completion of this
//      command.  phRobots will contain handles to robots connected to
//      this system. pdwNumRobots will will be updated with the number of
//      robots found.
//      Also, note that the hDrives[] array in the PTRobotInfo will not be
//      valid until PTRobot_EnumDrives is called.
//
//    Return:
//          PTROBOT_OK if Successful
//          PTROBOT_INVALID_ROBOT if no robots found
//          PTROBOT_SEQUENCE if this command is called out of sequence
//          PTROBOT_INTERNAL if an internal error occurred
//          PTROBOT_OVERFLOW if the number of robots found is > the value in
//                      pdwNumRobots
//
///////////////////////////
DWORD WINAPI PTRobot_EnumRobots(HANDLE * phRobots, DWORD * pdwNumRobots);
```

2.1.5 PTRobot_EnumDrives

```
///////////////////////////
//
//    PTRobot_EnumDrives
//
//    Description:
//          Function to enumerate the drives on the system and
//          determine which drives are under the control of this
//          robot.
//    Params:
//          hRobot      Handle to the robot to enumerate the drives of.
//          phDrives    points to an array of DWORDS to store
//                      the Drives found.
//          pdwNumDrives      points to a DWORD containing the number of HANDLEs
//                            in the phDrives array.  This value is an input
//                            and an output.  The user should specify the size
//                            (number of HANDLEs) of the phDrives array on input.
//                            The value of the pdwNumDrives on output will be the
//                            number of drives found.
//
//    Notes:
//          Both params will be updated upon successful completion of this
//          command.  phDrives will contain handles to drives connected to
//          this system. pdwNumDrives will will be updated with the number of
//          drives found.
//
//          The format of the drive handles is the following:
//
//          The least significant byte should contain the drive letter, the
//          other three bytes should contain the SCSI triple.
//          The drive can be identified by either of these methods.
//
//                For Example:  0x01030044 would identify a drive with:
//                              Host=1, ID = 3, LUN = 0, and a drive letter of "D"
//
//                To identify the same drive the client could pass
//                down 0x01030000, 0x00000044, or 0x01030044.
//
//    Return:
//          PTROBOT_OK if Successful
//          PTROBOT_SEQUENCE if this command is called out of sequence or after
//                      PTRobot_EnumDrivesWithList
//          PTROBOT_INTERNAL if an internal error occurred
//          PTROBOT_INVALID_ROBOT if the robot handle is invalid
//          PTROBOT_OVERFLOW if the number of drives found is > the value in
//                      pdwNumDrives
//          PTROBOT_MULTDRIVES if the module cannot determine which drives are
//                              robotically controlled. The calling application
//                              needs to use PTRobot_SetRoboticDrive to resolve
//                              this error.
//
///////////////////////////
DWORD WINAPI PTRobot_EnumDrives(HANDLE hRobot, HANDLE * phDrives, DWORD * pdwNumDrives);
```

2.1.6 PTRobot_EnumDrivesWithList

```
///////////////////////////
//
//    PTRobot_EnumDrivesWithList
//
//    Description:
//          Function to pass down drives enumerated by the calling app for
//          PTRobot to use in determining which drives are robotically controlled.
//          This is an alternative function to PTRobot_EnumDrives.
//    Params:
//          hRobot      Handle to the robot.
//          phDrives    points to an array of HANDLEs that contains the
//                      drive handles of the drives in the system
//          pdwNumDrives      points to a DWORD containing the number of HANDLEs
//                            in the phDrives array.
//          phRobotDrives     points to an array of HANDLEs that contains the
//                            drive handles of the drives contained in this robot.
//          pdwNumRobotDrives points to a DWORD containing the number of drives
//                            in the phRobotDrives array.
//
//    Notes:
//          phRobotDrives and pdwNumRobotDrives will be updated upon successful
//          completion of this command.  phRobotDrives will contain
//          handles to drives contained in the robot. pdwNumRobotDrives will be
//          updated with the number of drives found.
//
//          The format of the drive handle is the following:
//
//          The least significant byte should contain the drive letter, the
//                other three bytes should contain the SCSI triple.  The drive can
//                be identified by either of these methods.
//
//                For Example:  0x01030044 would identify a drive with:
//                                  Host=1, ID = 3, LUN = 0, and a drive letter of "D"
//
//                To identify the same drive the client could pass down
//                0x01030000, 0x00000044, or 0x01030044.
//
//          This function should be called instead of PTRobot_EnumDrives if the
//          calling application wants to enumerate the drives and have PTRobot
//          select the Robotically controlled drives from the list the calling
//          application provides.
//
//    Return:
//          PTROBOT_OK if Successful
//          PTROBOT_SEQUENCE if this command is called out of sequence or after
//                                     PTRobot_EnumDrives
//          PTROBOT_INTERNAL if an internal error occurred
//          PTROBOT_INVALID_ROBOT if the robot handle is invalid
//          PTROBOT_OVERFLOW if the number of robotic drives found is > the value
//                                   in pdwDrives
//          PTROBOT_MULTDRIVES if the module cannot determine which drives are
//                                     robotically controlled.
//
///////////////////////////
DWORD WINAPI PTRobot_EnumDrivesWithList(HANDLE hRobot, HANDLE * phDrives,
                  DWORD * pdwNumDrives, HANDLE * phRobotDrives, DWORD *
pdwNumRobotDrives);
```

2.1.7 PTRobot_SetRoboticDrive

```
///////////////////////////
//
//    PTRobot_SetRoboticDrive
//
//    Description:
//          Function to set a drive's position within the duplicator when
//          the PTROBOT_MULTDRIVES error is returned from either of the
//          EnumDrives functions.
//    Params:
//          hRobots     Handle to the Robot.
//          hDrive            Handle to the Drive
//          dwColIndex  Index identifying the column that the drive is in.
//                            (0 based where 0 is the left-most column)
//          dwRowIndex  Index identifying the row that the drive is in.
//                            (0 based where 0 is the top row)
//    Notes:
//    Return:
//          PTROBOT_OK if Successful
//          PTROBOT_SEQUENCE if this command is called out of sequence
//          PTROBOT_INTERNAL if an internal error occurred
//          PTROBOT_INVALID_ROBOT if the robot handle is invalid
//          PTROBOT_INVALID_DRIVE if the drive handle is invalid
//          PTROBOT_INVALID_DRIVE_POSITION if column/row ids are invalid.
//
///////////////////////////
DWORD WINAPI PTRobot_SetRoboticDrive(HANDLE hRobot, HANDLE hDrive, DWORD dwColIndex,
                                                DWORD dwRowIndex);
```

2.1.8 PTRobot_SetOpenCloseFunction

```
///////////////////////////
//
//    PTRobot_SetOpenCloseFunction
//
//    Description:
//          Function to set a calling application provided drive open/close
//          function.
//    Params:
```

```
//          pvOpenClose       pointer to a function to open and close the drive.
//                                  (setting to NULL will cause non-callback open/close
//                                   to be used).
//    Notes:
//          This function allows the calling application to provide the drive
//          open/closing functionality through their recording engine.  If this
//          function is not called then the drive will be opened/closed via OS
//          calls.  The function pointed to by the pvOpenClose param should be
//          defined as follows:
//
//          void   OpenCloseDrive(DWORD hDrive, DWORD dwOpen);
//
//          Please see the "Drive Open/Close" definitions above for the dwOpen
//          param.
//
//    Return:
//          PTROBOT_OK if Successful
//          PTROBOT_SEQUENCE if this command is called out of sequence
//          PTROBOT_INTERNAL if an internal error occurred
//
///////////////////////////
DWORD WINAPI PTRobot_SetOpenCloseFunction(void * pvOpenClose);
```

2.1.9 PTRobot_SetRobotOptions

```
///////////////////////////
//
//    PTRobot_SetRobotOptions
//
//    Description:
//          Function to set the current robot options.
//    Params:
//          hRobot            Handle to the robot
//          dwRobotOptions    DWORD containing the options to set.
//                            See "Robot Options" defines above
//    Notes:
//          You should call PTRobot_GetRobotOptions to get the current Options
//          and then set the options you want to change prior to calling
//          this function.
//    Return:
//          PTROBOT_OK if Successful
//          PTROBOT_SEQUENCE if this command is called out of sequence
//          PTROBOT_INTERNAL if an internal error occurred
//          PTROBOT_INVALID_ROBOT if the robot handle is invalid
//          PTROBOT_UNSUPPORTED_OPTION if the option is unsupported on that robot
//
///////////////////////////
DWORD WINAPI PTRobot_SetRobotOptions(HANDLE hRobot, DWORD dwRobotOptions);
```

2.1.10 PTRobot_GetRobotOptions

```
///////////////////////////
//
//    PTRobot_GetRobotOptions
```

```
//
//    Description:
//          Function to get the current robot options.
//    Params:
//          hRobot              Handle to the robot
//          pdwRobotOptions     points to a DWORD.
//                              See "Robot Options" defines above
//    Notes:
//    Return:
//          PTROBOT_OK if Successful
//          PTROBOT_SEQUENCE if this command is called out of sequence
//          PTROBOT_INTERNAL if an internal error occurred
//          PTROBOT_INVALID_ROBOT if the robot handle is invalid
//
///////////////////////////
DWORD WINAPI PTRobot_GetRobotOptions(HANDLE hRobot, DWORD *pdwRobotOptions);
```

2.1.11 PTRobot_GetErrorString

```
///////////////////////////
//
//    PTRobot_GetErrorString
//
//    Description:
//          Function to get the error string for a specific system
//          error or PTRobot API Return error.
//
//    Params:
//          hRobot                  Handle to the robot (from EnumRobots)
//                                  (Use NULL only if no handle has been obtained yet)
//          dwErrorNum              System Error Number
//          pwszErrorString         Error string returned
//                                  (Note wide characters.  Calling application must
//                                  allocate this memory).
//          dwMaxLength             Length of buffer pointed to by pwszErrorString
//                                  (number of wide characters)
//          dwLanguage              Language of string to return (See "Languages"
//                                  definitions above)
//
//    Notes:
//          dwErrorNum can be either dwSystemError from PTRobotStatus structure (which is
//          returned by PTRobot_GetRobotStatus()) or dwErrorNum can be an error returned
//          from a PTRobot_xxxxx call (e.g. PTROBOT_INVALID_ROBOT).
//
//
//    Return:
//          PTROBOT_OK if Successful
//          PTROBOT_INVALID_LANG if language is invalid
//          PTROBOT_INVALID_ERROR if error is invalid
//          PTROBOT_INTERNAL if buffer is undersized, etc.
//
///////////////////////////
DWORD WINAPI PTRobot_GetErrorString(HANDLE hRobot, DWORD dwErrorNum,
                                    WCHAR * pwszErrorString,  DWORD dwMaxLength,
                                    DWORD dwLanguage);
```

2.1.12 PTRobot_SetApplicationID

```
///////////////////////////
//
//    PTRobot_SetApplicationID
//
//    Description:
//          Function to set the Application ID.
//          The ID value is assigned for each application by Primera as needed.
//          Only applications that require special functionality will require this.
//          (note most applications will not need this).
//
//    Params:
//          dwAppID           Application ID specified by Primera
//
//    Notes:
//    Return:
//          PTROBOT_OK if Successful
//          PTROBOT_INTERNAL if an internal error occurred
//
///////////////////////////
 DWORD WINAPI PTRobot_SetApplicationID( DWORD dwAppID );
```

2.2 *PTRobot Info/Status Functions*

2.2.1 PTRobot_GetDriveInfo

```
///////////////////////////
//
//    PTRobot_GetDriveInfo
//
//    Description:
//          Function to get the drive info for a particular
//          drive handle.
//    Params:
//          hDrive      Handle to the drive (from EnumDrives)
//          pDrvInfo    points to a PTDriveInfo structure.
//    Notes:
//    Return:
//          PTROBOT_OK if Successful
//          PTROBOT_SEQUENCE if this command is called out of sequence
//          PTROBOT_INTERNAL if an internal error occurred
//          PTROBOT_INVALID_DRIVE if the drive handle is invalid
//
///////////////////////////
DWORD WINAPI PTRobot_GetDriveInfo(HANDLE hDrive, PTDriveInfo* pDrvInfo);
```

2.2.2 PTRobot_GetRobotInfo

```
///////////////////////////
//
//    PTRobot_GetRobotInfo
//
```

```
//    Description:
//          Function to get the robot info for a particular
//          robot handle.
//    Params:
//          hRobot      Handle to the robot (from EnumRobots)
//          pRobotInfo  points to a PTRobotInfo structure.
//    Notes:
//    Return:
//          PTROBOT_OK if Successful
//          PTROBOT_SEQUENCE if this command is called out of sequence
//          PTROBOT_INTERNAL if an internal error occurred
//          PTROBOT_INVALID_ROBOT if the robot handle is invalid
//
///////////////////////////
DWORD WINAPI PTRobot_GetRobotInfo(HANDLE hRobot, PTRobotInfo *pRobotInfo);
```

2.2.3 PTRobot_GetRobotStatus

```
///////////////////////////
//
//    PTRobot_GetRobotStatus
//
//    Description:
//          Function to get the current status for a particular
//          robot.
//    Notes: Do NOT call in too tight of a loop (e.g. do not call more often
//           than every 500ms or so).
//    Params:
//          hRobot            Handle to the robot (from EnumRobots)
//          pRobotStatus      points to a PTRobotStatus structure.
//    Notes:
//    Return:
//          PTROBOT_OK if Successful
//          PTROBOT_SEQUENCE if this command is called out of sequence
//          PTROBOT_INTERNAL if an internal error occurred
//          PTROBOT_INVALID_ROBOT if the robot handle is invalid
//
///////////////////////////
DWORD WINAPI PTRobot_GetRobotStatus(HANDLE hRobot, PTRobotStatus *pRobotStatus);
```

2.2.4 PTRobot_GetMediaInfo

```
///////////////////////////
//
//    PTRobot_GetMediaInfo
//
//    Description:
//          This function will get information on the media that
//          is loaded in the drive.
//    Params:
//          hDrive                  Handle to the drive (from EnumDrives)
//          PTMediaInfo *           points to Media info structure (see section 3.5)
//                                  (the structure will be filled in if successful)
//    Notes:
//    Return:
```

```
//
//          PTROBOT_OK if successful and media is found and the media is valid.
//      PTROBOT_INVALID_MEDIA if the media is not valid
//      PTROBOT_NO_MEDIA if no media is found
//      PTROBOT_INVALID_DRIVE if the drive is not valid
//      PTROBOT_INTERNAL some other error
//
///////////////////////////
DWORD WINAPI PTRobot_GetMediaInfo(HANDLE hDrive, PTMediaInfo * pDiscInfo );
```

2.2.5 **PTRobot_GetRobotInfo2**

```
///////////////////////////
//
//    PTRobot_GetRobotInfo2
//
//    Description:
//          Function to get ADDITIONAL robot info for a particular
//          robot handle.
//    Params:
//          hRobot            Handle to the robot (from EnumRobots)
//          pRobotInfo2       points to a PTRobotInfo structure.
//    Notes:
//    Return:
//          PTROBOT_OK if Successful
//          PTROBOT_SEQUENCE if this command is called out of sequence
//          PTROBOT_INTERNAL if an internal error occurred
//          PTROBOT_INVALID_ROBOT if the robot handle is invalid
//
///////////////////////////
DWORD WINAPI PTRobot_GetRobotInfo2(HANDLE hRobot, PTRobotInfo2 *pRobotInfo2);
```

2.2.6 **PTRobot_GetRobotStatus2**

```
///////////////////////////
//
//    PTRobot_GetRobotStatus2
//
//    Description:
//          Function to get the Additional current status for a particular
//          robot.
//    Notes: Do NOT call in too tight of a loop (e.g. do not call more often
//           than every 500ms or so).
//    Params:
//          hRobot                  Handle to the robot (from EnumRobots)
//          pRobotStatus2     points to a PTRobotStatus2 structure.
//    Notes:
//    Return:
//          PTROBOT_OK if Successful
//          PTROBOT_SEQUENCE if this command is called out of sequence
//          PTROBOT_INTERNAL if an internal error occurred
//          PTROBOT_INVALID_ROBOT if the robot handle is invalid
//          PTROBOT_BUSY if no response from robot
```

```
//
///////////////////////////
DWORD WINAPI PTRobot_GetRobotStatus2(HANDLE hRobot, PTRobotStatus2 *pRobotStatus2);
```

2.2.7 PTRobot_GetManufactureInfo

```
///////////////////////////
//
//    PTRobot_GetManufactureInfo
//
//    Description:
//          Function to get manufacture info
//    Params:
//          hRobot                  Handle to the robot (from EnumRobots)
//          pPTManufactureInfo      points to a PTManufactureInfo structure.
//    Notes:
//
//    Return:
//          PTROBOT_OK if Successful (and fills in pPTManufactureInfo)
//          PTROBOT_INTERNAL if an internal error occurred
//          PTROBOT_INVALID_ROBOT if invalid robot specified
//
///////////////////////////
DWORD WINAPI PTRobot_GetManufactureInfo( HANDLE hRobot, PTManufactureInfo *
                                                            pManufactureInfo);
```

2.3 PTRobot Robotic Functions

2.3.1 PTRobot_LoadDrive

```
///////////////////////////
//
//    PTRobot_LoadDrive
//
//    Description:
//          Function to load a drive from an input location
//    Params:
//          hRobot            Handle to the robot (from EnumRobots)
//          hDrive            Handle to the drive (from EnumDrives)
//          dwFromLocation    DWORD containing the "from" location
//                LOCATION_AUTO = Automatically choose the bin
//                1 = Bin1 (right-most bin)
//                2 = Bin2
//                ...
//                LOCATION_PRINTER = Printer
//          dwClearDrive      Clear drive before loading.
//                                  (See Clear Drive section)
//
//    Notes:
//          Clear drive before loading should be done the first loading.  This will
//          cause the picker to attempt to pick discs out of the drive to determine
//          if any discs were left in the drive from a previous job.
//    Notes:
//    Return:
//          PTROBOT_OK if Successful
//          PTROBOT_SEQUENCE if this command is called out of sequence
//          PTROBOT_INTERNAL if an internal error occurred
//          PTROBOT_INVALID_ROBOT if the robot handle is invalid
//          PTROBOT_INVALID_DRIVE if the drive handle is invalid
//          PTROBOT_INVALID_LOCATION if the location is invalid
//
///////////////////////////
DWORD WINAPI PTRobot_LoadDrive(HANDLE hRobot, HANDLE hDrive, DWORD dwFromLocation, DWORD
dwClearDrive);
```

2.3.2 PTRobot_LoadPrinter

```
///////////////////////////
//
//    PTRobot_LoadPrinter
//
//    Description:
//          Function to load the printer from an input bin location
//    Params:
//          hRobot              Handle to the robot (from EnumRobots)
//          dwFromLocation      DWORD containing the "from" location
//                              LOCATION_AUTO = Automatically choose the bin
//                              1 = Bin1 (right-most bin)
//                              2 = Bin2
//                              ...
//
//    Notes:
//    Return:
//          PTROBOT_OK if Successful
//          PTROBOT_SEQUENCE if this command is called out of sequence
//          PTROBOT_INTERNAL if an internal error occurred
//          PTROBOT_INVALID_ROBOT if the robot handle is invalid
//          PTROBOT_NO_PRINTER if the robot doesn't have a printer
//          PTROBOT_INVALID_LOCATION if the location is invalid
//
///////////////////////////
DWORD WINAPI PTRobot_LoadPrinter(HANDLE hRobot, DWORD dwFromLocation);
```

2.3.3 PTRobot_LoadPrinterFromDrive

```
///////////////////////////
//
//    PTRobot_LoadPrinterFromDrive
//
//    Description:
//          Function to load the printer from a drive
//    Params:
//          hRobot              Handle to the robot (from EnumRobots)
//          hDrive              Handle to the drive (from EnumDrives)
//    Notes:
//    Return:
//          PTROBOT_OK if Successful
//          PTROBOT_SEQUENCE if this command is called out of sequence
//          PTROBOT_INTERNAL if an internal error occurred
//          PTROBOT_INVALID_ROBOT if the robot handle is invalid
//          PTROBOT_INVALID_DRIVE if the drive handle is invalid
//          PTROBOT_NO_PRINTER if the robot doesn't have a printer
//
///////////////////////////
DWORD WINAPI PTRobot_LoadPrinterFromDrive(HANDLE hRobot, HANDLE hDrive);
```

2.3.4 PTRobot_UnLoadDrive

```
///////////////////////////
//
//    PTRobot_UnLoadDrive
//
//    Description:
//          Function to unload the drive to an output position.
//    Params:
//          hRobot            Handle to the robot (from EnumRobots)
//          hDrive            Handle to the drive (from EnumDrives)
//          dwToLocation      DWORD containing the "to" location
//                LOCATION_AUTO = Automatically choose the bin
//                1 = Bin1 (right-most bin)
//                2 = Bin2
//                ...
//                LOCATION_REJECT = Reject
//
//    Notes:
//    Return:
//          PTROBOT_OK if Successful
//          PTROBOT_SEQUENCE if this command is called out of sequence
//          PTROBOT_INTERNAL if an internal error occurred
//          PTROBOT_INVALID_ROBOT if the robot handle is invalid
//          PTROBOT_INVALID_DRIVE if the drive handle is invalid
//          PTROBOT_INVALID_LOCATION if the location is invalid
//
///////////////////////////
DWORD WINAPI PTRobot_UnLoadDrive(HANDLE hRobot, HANDLE hDrive, DWORD dwToLocation);
```

2.3.5 PTRobot_UnLoadPrinter

```
///////////////////////////
//
//    PTRobot_UnLoadPrinter
//
//    Description:
//          Function to unload the printer to an output position.
//    Params:
//          hRobot            Handle to the robot (from EnumRobots)
//          dwToLocation      DWORD containing the "to" location
//                  LOCATION_AUTO = Automatically choose the bin
//                  1 = Bin1 (right-most bin)
//                  2 = Bin2
//                  ...
//                  LOCATION_REJECT = Reject
//
//      Notes:
//      Return:
//            PTROBOT_OK if Successful
//            PTROBOT_SEQUENCE if this command is called out of sequence
//            PTROBOT_INTERNAL if an internal error occurred
//            PTROBOT_INVALID_ROBOT if the robot handle is invalid
//            PTROBOT_NO_PRINTER if the robot doesn't have a printer
//            PTROBOT_INVALID_LOCATION if the location is invalid
//
//
///////////////////////////
DWORD WINAPI PTRobot_UnLoadPrinter(HANDLE hRobot, DWORD dwToLocation);
```

2.3.6 PTRobot_MoveDiscBetweenLocations

```
///////////////////////////
//
//      PTRobot_MoveDiscBetweenLocations
//
//      Description:
//            Function to move disc from one bin to another bin
//      Params:
//            hRobot                  Handle to the robot (from EnumRobots)
//            dwFromLocation    DWORD containing the from location
//                  1 = Bin1 (right-most bin)
//                  2 = Bin2
//                  ...
//            dwToLocation      DWORD containing the "to" location
//                  1 = Bin1 (right-most bin)
//                  2 = Bin2
//                  ...
//                  LOCATION_REJECT = Reject
//
//      Notes:
//      Return:
//            PTROBOT_OK if Successful
//            PTROBOT_SEQUENCE if this command is called out of sequence
//            PTROBOT_INTERNAL if an internal error occurred
//            PTROBOT_INVALID_ROBOT if the robot handle is invalid
//            PTROBOT_INVALID_LOCATION if the location is invalid
//
///////////////////////////
DWORD WINAPI PTRobot_MoveDiscBetweenLocations(HANDLE hRobot, DWORD dwFromLocation, DWORD
dwToLocation)
```

2.3.7 PTRobot_PrintFile

```
///////////////////////////
//
//    PTRobot_PrintFile
//
//    Description:
//          Function to print a Surething image (.STD), raster image (.JPG, .BMP, .TIF,
//          etc.), or  .PRN file to the printer.
//    Params:
//          hRobot            Handle to the robot (from EnumRobots)
//          tszFile           File to print (.STD, .PRN, .JPG, .BMP)
//          dwPrintIndex      Print index for multiple print jobs.
//    Notes:
//          The dwPrintIndex is used when printing an .STD file with merge fields.  This
//          value represents which merge record to use for this print.
//    Return:
//          PTROBOT_OK if Successful
//          PTROBOT_SEQUENCE if this command is called out of sequence
//          PTROBOT_INTERNAL if an internal error occurred
//          PTROBOT_INVALID_ROBOT if the robot handle is invalid
//          PTROBOT_NO_PRINTER if the robot doesn't have a printer
//          PTROBOT_PRN_INVALID if the prn file is not valid for the printer
//          PTROBOT_PRINTFILE_NOT_FOUND if the file doesn't exist
//          PTROBOT_PRINTAPP_NOT_INSTALLED if the required print application is not
//                installed.
//
///////////////////////////
DWORD WINAPI PTRobot_PrintFile(HANDLE hRobot, TCHAR * tszFile, DWORD dwPrintIndex);
```

2.3.8 PTRobot_PrintFileWithMerge

```
///////////////////////////
//
//    PTRobot_PrintFileWithMerge
//
//    Description:
//          Function to print a Surething .STD file that has Merge Text/Photos.
//          The Merge Text and/or Photos can be specified in a variable number of
//          arguments passed into this function.  The Surething file should be designed
//          with the same number of merge strings passed in here.
//
//    Params:
//          hRobot                  Handle to the robot (from EnumRobots)
//          tszFile                 Surething (.STD) File to print.
//          dwNumMergeStrings       Number of merge strings to follow
//          ...                     Variable number of pointers to TCHAR strings
//                                  These are the merge strings or photo names (including
//                                  path) to be printed.
//                                  NOTE: For the strings that follow dwMergeStrings to
//                                  be used, the user must have “Set Merge File” within
//                                  the .STD file
//                                  ** Limit each string to 256 characters or less **
//
//    Return:
```

```
//          PTROBOT_OK if Successful
//          PTROBOT_SEQUENCE if this command is called out of sequence
//          PTROBOT_INTERNAL if an internal error occurred
//          PTROBOT_INVALID_ROBOT if the robot handle is invalid
//          PTROBOT_NO_PRINTER if the robot doesn't have a printer
//          PTROBOT_PRINTFILE_NOT_FOUND if the file doesn't exist
//          PTROBOT_PRINTAPP_NOT_INSTALLED if the required print application is not
//                installed.
//          PTROBOT_PRINTFILE_INVALID if the filename is not .STD
//
///////////////////////////
DWORD WINAPI PTRobot_PrintFileWithMerge(HANDLE hRobot,
                                         TCHAR * tszFile,
                                         DWORD dwNumMergeStrings,
                                         ...);
```

2.3.9 PTRobot_SetPrinterSettings

```
///////////////////////////
//
//    PTRobot_SetPrinterSettings
//
//    Description:
//          Function to set some printer driver settings
//    Params:
//          hRobot                  Handle to the robot (from EnumRobots)
//          pPrinterSettings        points to a PTPrinterSettings structure.
//
//    Notes:
//          If this function is not called the default print settings will be used. This
//          function will change the system default print settings.
//          Starting with Version 1.2.0 the system default print settings will be
//          restored after calling PTRobot_PrintFile() or PTRobot_PrintFileWithMerge().
//    Return:
//          PTROBOT_OK if Successful
//          PTROBOT_SEQUENCE if this command is called out of sequence
//          PTROBOT_INTERNAL if an internal error occurred
//          PTROBOT_INVALID_ROBOT if the robot handle is invalid
//          PTROBOT_NO_PRINTER if the robot doesn't have a printer
//          PTROBOT_INVALID_PRINTER_SETTINGS if the printer settings are invalid
//
///////////////////////////
DWORD WINAPI PTRobot_SetPrinterSettings(HANDLE hRobot, PTPrinterSettings
*pPrinterSettings);
```

2.3.10 PTRobot_GetPrinterSettings

```
///////////////////////////
//
//    PTRobot_GetPrinterSettings
//
//    Description:
//          Function to get some printer driver settings
//    Params:
//          hRobot                  Handle to the robot (from EnumRobots)
//          pPrinterSettings        points to a PTPrinterSettings structure.
//
//    Notes:
//          If this function is not called the default print settings will be used.
//    Return:
//          PTROBOT_OK if Successful
//          PTROBOT_SEQUENCE if this command is called out of sequence
//          PTROBOT_INTERNAL if an internal error occurred
//          PTROBOT_INVALID_ROBOT if the robot handle is invalid
//          PTROBOT_NO_PRINTER if the robot doesn't have a printer
//
///////////////////////////
DWORD WINAPI PTRobot_GetPrinterSettings(HANDLE hRobot, PTPrinterSettings
*pPrinterSettings);
```

2.3.11 PTRobot_KillSystemError

```
///////////////////////////
//
//    PTRobot_KillSystemError
//
//    Description:
//          Function to kill a system error.
//    Params:
//          hRobot             Handle to the robot (from EnumRobots)
//          dwResetPrinter     DWORD to notify if the printer should be reset.
//                             1 = Reset the printer
//                             0 = do not reset the printer
//
//    Notes:
//          If there is no system error and dwResetPrinter is set to 1 the
//          printer will be reset.  Otherwise if there is a system error, that error
//          will be cleared (if possible) and the printer will be reset if the
//          dwResetPrinter is set to 1.
//    Return:
//          PTROBOT_OK if Successful
//          PTROBOT_SEQUENCE if this command is called out of sequence
//          PTROBOT_INTERNAL if an internal error occurred
//          PTROBOT_INVALID_ROBOT if the robot handle is invalid
//
///////////////////////////
DWORD WINAPI PTRobot_KillSystemError(HANDLE hRobot, DWORD dwResetPrinter);
```

2.3.12 PTRobot_SystemAction

```
///////////////////////////
//
//    PTRobot_SystemAction
//
//    Description:
//          Function to instruct the system to perform a specifc action.
//    Params:
//          hRobot       Handle to the robot (from EnumRobots)
//          dwAction     Action to perform
```

```
//
//    Notes:
//          This function is used to perform a specific function on a
//          robot.  The defined actions and their descriptions are detailed
//          below.
//
//          Action:
//                PTACT_ALIGNPRINTER -> Align the Printer (Disc Publisher PRO only)
//          Description:
//                This will cause an alignment print to occur on the printer and
//                this function will return when the alignment is complete.
//
//          Action:
//              PTACT_IGNOREINKLOW -> Ignore Ink Low (Disc Publisher PRO only)
//          Description:
//                This will cause an ink low system error to be ignored.
//
//          Action:
//              PTACT_DISABLEPWRBUTTON -> Disable Power Button
//          Description:
//                This will disable the power button on Disc Publisher II and PRO.
//
//          Action:
//              PTACT_REINIT_DRIVES -> Re-initialize drives
//          Description:
//                PTRobot maintains Registry values for persistent settings including
//                drive serial numbers.  This action will clear the drive serial numbers
//                stored which will force the user to re-identify the robotically
//                controlled drives.
//
//          Action:
//              PTACT_IDENTIFY -> Identify a robot
//          Description:
//                This will cause the robot to do something to visually identify itself
//                For example the Bravo units will move their printer tray.
//    Return:
//          PTROBOT_OK if Successful
//          PTROBOT_SEQUENCE if this command is called out of sequence
//          PTROBOT_INTERNAL if an internal error occurred
//          PTROBOT_INVALID_ROBOT if the robot handle is invalid
//          PTROBOT_INVALID_ACTION if the robot action is invalid
//
///////////////////////////
DWORD WINAPI PTRobot_SystemAction(HANDLE hRobot, DWORD dwAction);
```

2.3.13 PTRobot_OpenCloseDrive

```
///////////////////////////
//
//    PTRobot_OpenCloseDrive
//
//    Description:
//          Function to open or close a drive
//    Params:
//          hDrive      Handle to the drive (from EnumDrives)
//          dwOpen      See (Drive Open/Close) section above
//                      (DRIVE_OPEN=0  DRIVE_CLOSE=1)
//
//    Notes:
//    Return:
//          PTROBOT_OK if Successful
//          PTROBOT_SEQUENCE if this command is called out of sequence
//          PTROBOT_INTERNAL if an internal error occurred
//          PTROBOT_INVALID_DRIVE if the drive handle is invalid
//
///////////////////////////
DWORD WINAPI PTRobot_OpenCloseDrive(HANDLE hDrive, DWORD dwOpen );
```

2.3.14 PTRobot_PrintFileWithMerge2

```
///////////////////////////////////////
//
//    PTRobot_PrintFileWithMerge2
//
//    Description:
//          Function to print a Surething .STD file that has Merge Text/Photos.
//          The Merge Text and/or Photos MUST be specified BEFORE THIS CALL, by
//          calling PTRobot_AddMergeFields().  PTRobot_AddMergeFields() must be called
//          once for every Merge field that is designed into the SureThing file.
//          Then, this function is called to print the file with the specified
//          Merge data.
//
//    Params:
//          hRobot            Handle to the robot (from EnumRobots)
//          tszFile           Surething (.STD) File to print.
//          fClearMergeList   Whether or not to clear the list of merge strings
//                            (or photo names) that were stored from previous calls
//                            to PTRobot_AddMergeFields().  If doing mutiple discs with
//                            the same merge data then set to FALSE.
//                            If merge data will be changing for each disc then
//                            set to TRUE.
//
//    NOTE: For this to work, the user must have “Set Merge File” within the .STD file
//
//    Return:
//      PTROBOT_OK if Successful
//          PTROBOT_SEQUENCE if this command is called out of sequence
//          PTROBOT_INTERNAL if an internal error occurred
//          PTROBOT_INVALID_ROBOT if the robot handle is invalid
//          PTROBOT_NO_PRINTER if the robot doesn't have a printer
//          PTROBOT_PRINTFILE_NOT_FOUND if the file doesn't exist
//          PTROBOT_PRINTAPP_NOT_INSTALLED if the required print application is not
//                installed.
//          PTROBOT_PRINTFILE_INVALID if the filename is not .STD
//
///////////////////////////
DWORD WINAPI PTRobot_PrintFileWithMerge2(HANDLE hRobot,
                                                      TCHAR * tszFile,
                                                      BOOL fClearMergeList);
```

2.3.15 PTRobot_SetPrinterSettings2

```
///////////////////////////
//
//    PTRobot_SetPrinterSettings2
//
//    Description:
//          Function to set additional printer driver settings
//    Params:
//          hRobot                      Handle to the robot (from EnumRobots)
//          pPrinterSettings2 points to a PTPrinterSettings2 structure.
//
//    Notes:
//          If this function is not called the default print settings will be used. This
//      function will change the system default print settings.
//          As of Version 1.2.0 the system default print settings will restored after a
print
//          is sent via PTRobot_PrintFile() or PTRobot_PrintFileWithMerge().
//
//    Return:
//      PTROBOT_OK if Successful
//          PTROBOT_SEQUENCE if this command is called out of sequence
//          PTROBOT_INTERNAL if an internal error occurred
//          PTROBOT_INVALID_ROBOT if the robot handle is invalid
//          PTROBOT_NO_PRINTER if the robot doesn't have a printer
//          PTROBOT_INVALID_PRINTER_SETTINGS if the printer settings are invalid
//
///////////////////////////
DWORD WINAPI PTRobot_SetPrinterSettings2(HANDLE hRobot, PTPrinterSettings2
                                                              *pPrinterSettings2);
```

2.3.16 PTRobot_GetPrinterSettings2

```
///////////////////////////
//
//    PTRobot_GetPrinterSettings2
//
//    Description:
//          Function to get some additional printer driver settings
//    Params:
//          hRobot                      Handle to the robot (from EnumRobots)
//          pPrinterSettings2 points to a PTPrinterSettings2 structure.
//
//    Notes:
//          If this function is not called the default print settings will be used.
//    Return:
//      PTROBOT_OK if Successful
//          PTROBOT_SEQUENCE if this command is called out of sequence
//          PTROBOT_INTERNAL if an internal error occurred
//          PTROBOT_INVALID_ROBOT if the robot handle is invalid
//          PTROBOT_NO_PRINTER if the robot doesn't have a printer
//
///////////////////////////
DWORD WINAPI PTRobot_GetPrinterSettings2(HANDLE hRobot, PTPrinterSettings2
                                                                    *pPrinterSettings2);
```

2.4 PTRobot Misc Functions

2.4.1 PTRobot_GetSureThingPreview

```
///////////////////////////
//
//    PTRobot_GetSureThingPreview
//
//    Description:
//          Function to get a preview of a SureThing file
//    Params:
//          tszSureThingFile  The SureThing file to get a preview of
//          tszOutputFile     The file name (including path) of desired output file
//                            (NOTE: must have extension of .JPG, .BMP, or .PNG)
//          dwResolution      Resolution (in DPI) of output file (Valid values: 50-600)
//    Notes:
//          1)This function returns immediately, but the output file may take several
//            seconds to generate. Caller should keep trying to get exclusive read access
//            to the output file.
//          2)The output file is NOT deleted.  Caller is responsible for deleting,
//            if desired.
//          3) Calling with dwResolution=0 is special case that will return
//                PTROBOT_PRINTAPP_NOT_INSTALLED if SureThing is not installed, otherwise it
//                will return PTROBOT_OK.  No Preview will be generated.
//
//    Return:
//          PTROBOT_OK if Successful
//          PTROBOT_PRINTFILE_INVALID if fails to generate preview
//          PTROBOT_INVALID_EXTENSION if not valid output file extension(.JPG,.BMP,.PNG)
//
///////////////////////////
DWORD WINAPI PTRobot_GetSureThingPreview(TCHAR * tszSureThingFile,
                                          TCHAR * tszOutputFile,
                                          DWORD dwResolution);
```

2.4.2 PTRobot_AddMergeFields

```
///////////////////////////
//
//    PTRobot_AddMergeFields
//
//    Description:
//          This function is used in conjuction with PTRobot_PrintMergeFile2()
//          to print a SureThing file with merge/replaceable text fields (or photos).
//          DESIGN TIME:
//          User first designs a SureThing (.STD) file with replaceable Text and/or
//          photos and then sets the Merge file (using Tools/Set Merge File).
//          RUN TIME:
//          This function is called repeatedly, passing in the text you want printed
//          (and/or the name & path of the photo you want printed).
//          The number of times this function is called should be equal to the number of
//          Merge Fields inserted at design time.
//          Then, call PTRobot_PrintFileWithMerge2() to print the SureThing file, using
//          the merge fields set with this call.
//
//    Params:
//
//    Notes:
//    Return:
//          PTROBOT_OK if Successful
//
//
///////////////////////////
DWORD WINAPI PTRobot_AddMergeFields(HANDLE hRobot, const TCHAR * tszField );
```

2.4.3 **PTRobot_ClearMergeList**

```
///////////////////////////
//
//    PTRobot_ClearMergeList
//
//    Description:
//          Function to clear the List of Merge strings (or photo names)
//          that were stored from previous calls to PTRobot_AddMergeFields().
//          This function would be used if doing a print preview of a STD file
//          that has merge fields (ie. using PTRobot_GetSureThingPreview() ).
//          This function is NOT needed if only printing a SureThing file with merge
//          fields because PTRobot_PrintFileWithMerge2() has a flag to clear the merge
//          list after printing.
//          So, the use of this function would be like this:
//                PTRobot_AddMergeFields()
//                PTRobot_AddMergeFields()...
//                PTRobot_GetSureThingPreview()
//                PTRobot_ClearMergeList()
//
//          NOTE: For this to work, the user must have “Set Merge File”
//          within the .STD file
//
//    Params:
//          hRobot                        Handle to the robot (from EnumRobots)
//    Return:
//          PTROBOT_OK
//
///////////////////////////
DWORD WINAPI PTRobot_ClearMergeList( HANDLE hRobot );
```

2.4.4 **PTRobot_SetPrintCopies**

```
///////////////////////////
//
```

```
//    PTRobot_SetPrintCopies
//
//    Description:
//          This function is used for a multiple auto-print job.
//    This function is called each time prior to calling one of the print functions
//  (e.g. PTRobot_PrintFile() ).  The number of copies is set back to 1 after printing
//  Not all robots support this feature (PTACT_AUTOPRINTER_MODE specifies support)
//    e.g. Bravo 4100 prints mulitple copies faster with this method.
//
//
//    Params:
//
//    Notes:
//    Return:
//          PTROBOT_OK if Successful
//
//
///////////////////////////
DWORD WINAPI PTRobot_SetPrintCopies(HANDLE hRobot, DWORD dwCopies );
```

3 Type Definitions

NOTE: As of version 1.3.0 there are two different versions of PTRobot:
Multibyte character set (MCBS/ANSI) and Unicode.
In the Unicode version, a TCHAR is a 2-byte wide character. This is needed for 2-byte languages such as Chinese, Japanese, Korean, etc.
In the MCBS version, a TCHAR is a 1-byte character.

3.1 *PTDriveInfo Structure*

```
typedef struct
{
      HANDLE hDrive;                  //Drive Handle.
      TCHAR tszDriveName[132];        //Drive String (reported from drive)
      TCHAR tszFirmwareVer[40];       //Drive FW version
      TCHAR tszSerialNum[40];         //Drive Serial Number
      HANDLE hRobot;
      DWORD dwDriveColumn;            //Drive Column (0 based - 0 is leftmost column)
      DWORD dwDriveRow;               //Drive Row (0 based - 0 is the top row)
}PTDriveInfo, *pPTDriveInfo;
```

3.2 *PTRobotInfo Structure*

```
typedef struct
{
      HANDLE hRobot;                 //Robot Handle
      TCHAR tszRobotDesc[100];       //Robot Description
      DWORD dwRobotType;             //See "Robot Type" section 4.4
      DWORD dwNumDrives;             //Number of Recorders on this robot
      DWORD dwNumPrinters;           //Number of Printers on this robot (0 or 1)
      DWORD dwNumBins;               //Number of Bins on this robot
      DWORD dwDriveColumns;          //Number of Drive Columns
      DWORD dwDriveRows;             //Number of Drive Rows
      TCHAR tszRobotFirmware[20];    //String Containing the FW Version of the Robot
      DWORD dwOptions;               //See "Robot Options" section 4.6
      DWORD dwAction;                //See "Robot Actions" section 4.7
      HANDLE hDrives[10];
      DWORD dwDriveBusType;          //BusType of the Drives
}PTRobotInfo, *pPTRobotInfo;
```

3.3 *PTRobotInfo2 Structure*

```
typedef struct
{
      DWORD dwNumCartridges;         //Max Number of cartridges robot can hold
      DWORD dwCartridgeType[8];      // First element is left-most cartridge and last
                                     // element is right-most cartridge (from the user's
                                    // viewpoint). see "Cartridge Types" above
      DWORD dwFirmware2Code;
      DWORD dwPGA;
      DWORD dwModel;
      DWORD dwUSBSerialNum;
      DWORD dwMaxDiscsPerBin;
      DWORD dwReserved[9];          // reserved for future data

}PTRobotInfo2, *pPTRobotInfo2;
```

3.4 PTRobotStatus Structure

```
typedef struct
{
      DWORD dwSystemState;            //See "System State" section 4.3
      DWORD dwSystemError;            //See "System Error" section 4.2
      DWORD dwCurrColorSpits;
      DWORD dwCurrBlackSpits;
      DWORD dwFullColorSpits;
      DWORD dwFullBlackSpits;
}PTRobotStatus, *pPTRobotStatus;
```

3.5 PTPrinterSettings Structure

```
typedef struct
{
      DWORD dwPrintQuality;           //See "Print Quality" section 4.9
      DWORD dwInnerDiam;              //units in .1mm increments (150 - 500)
      DWORD dwOuterMargin;            //units in .1mm increments (0 - 20)
}PTPrinterSettings,  *pPTPrinterSettings;
```

3.6 PTPrinterSettings2 Structure

```
typedef struct
{
      // NOTE: 0xffff is a special value depending on Get or Set--
      // Get:  0xffff means that the setting is not supported
      // Set:  0xffff means to use the current driver setting
      DWORD dwPrintQuality;   //See "Print Quality" section above
      DWORD dwInnerDiam;      //units in .1mm increments (150 - 500)
      DWORD dwOuterMargin;    //units in .1mm increments (0 - 20)
      DWORD dwCartridge;      // Cartridge type to use.  1=Black  2=Color  3=Color+Black
      DWORD dwColorMatchType; // Color Rendering. 0=Best for Graphics  1=Best for Photos
      DWORD dwColorTable;     // Color Table number to use. Valid values: 1-6
      DWORD dwSaturation;     // Amount of saturation. 0-100 where 100 is full saturation
      DWORD dwPrintBidi;      // Print bi-directionally or not.  0=No  1=Yes
      DWORD dwRotate180;      // Rotate the image 180 degrees or not.  0=No  1=Yes
      DWORD dwKioskPrintOnly; // Kiosk mode when doing print-only.  0=No  1=Yes
      DWORD dwReserved[9];
}PTPrinterSettings2,  *pPTPrinterSettings2;
```

3.7 PTMediaInfo Structure

```
typedef struct
{
    TCHAR tszMediaID[20];
    TCHAR tszMediaType[20];
} PTMediaInfo, *pPTMediaInfo;
```

3.8 PTRobotInfo2 Structure

```
typedef struct
{
      DWORD dwNumCartridges;        //Max Number of cartridges robot can hold
      DWORD dwCartridgeType[8];     // First element is left-most cartridge and last
                                    // element is right-most cartridge (from the user's
                                    // viewpoint) see "Cartridge Types" section 4.16
      DWORD dwFirmware2Code;
      DWORD dwPGA;
      DWORD dwModel;
      DWORD dwUSBSerialNum;
      DWORD dwMaxDiscsPerBin;
      DWORD dwReserved[9];          // reserved for future data
}PTRobotInfo2, *pPTRobotInfo2;
```

3.9 PTRoboStatus2 Structure

```
#define UNKNOWN_NUM_DISCS 255
typedef struct
{
      DWORD dwCartridgeTypes;             // see "Cartridges Installed" section 4.17
      DWORD dwNumDiscsInBins[5];          // 0th element is left-most bin (values are
                                          // 255 if unknown)
      DWORD dwTotalPrints;                // Total # of prints
      DWORD dwTotalPicks;                 // Total # of picks from input bin
      DWORD dwVerticalOffset;             // Vertical print offset (300dpi units)
      DWORD dwHorizontalOffset;           // Horizontal print offset (300dpi units)
      DWORD dwPrinterTrayStatus;          // See "Printer Tray Status" section 4.14
      DWORD dwDiscPickSwitchStatus;       // See "Disc Pick Switch Status" section 4.15
      DWORD dwCoverBeenOpenedFlag;        // set to 1 if cover has been opened
      DWORD dwCartridgeInstalled[8];      // 0=not installed.  1=valid cartridge.
                                                2=invalid cartridge
      DWORD dwCartridgeNeedsAlign[8];     // 1 if cartridge needs alignment
      DWORD dwSystemStateHW;              // actual system state as reported directly
                                          from the printer (may be different than
                                          dwSystemState in PTRobotStatus)
      long  lYellowInkRemaining;          // Yellow % remaining (in ten thousandths of a
                                                percent).  e.g. 891723 = 89.1723%
      long  lMagentaInkRemaining;         // Magenta % remaining (in ten thousandths of a
                                                percent).  e.g. 891723 = 89.1723%
      long  lCyanInkRemaining;            // Cyan % remaining (in ten thousandths of a
                                                percent).  e.g. 891723 = 89.1723%
      long  lBlackInkRemaining;           // Black % remaining (in ten thousandths of a
                                                percent).  e.g. 891723 = 89.1723%
                                          // -BELOW ONLY VALID FOR DISC PUBLISHER 4100-
      BYTE  bCartridgeStatusYellow;       // See CartridgeInfoType below for valid values
      BYTE  bCartridgeStatusMagenta;      // See CartridgeInfoType below for valid values
      BYTE  bCartridgeStatusCyan;         // See CartridgeInfoType below for valid values
      BYTE  bCartridgeStatusBlack;        // See CartridgeInfoType below for valid values
      DWORD dwReserved[6];                // reserved for future data
}PTRobotStatus2, *pPTRobotStatus2;
```

3.10 PTManufactureInfo Structure

```
typedef struct
{
      TCHAR tszSerialNum[11];
      TCHAR tszManufactureDate[12];
      DWORD dwFiller[20];
}PTManufactureInfo, *pPTManufactureInfo;
```

3.11 CartridgeInfoType enum

```
// Cartridge Status Information Values for bCartridgeStatus[] in PTRobotStatus2 above
    (for Bravo 4100)
typedef enum
{
   CARTRIDGE_INFO_STILL_READING,
   CARTRIDGE_INFO_UNUSED,
   CARTRIDGE_INFO_MISSING,
   CARTRIDGE_INFO_INVALID,
   CARTRIDGE_INFO_COMM_ERROR,
   CARTRIDGE_INFO_BAD_POSITION,
   CARTRIDGE_INFO_BAD_INSTALL,
   CARTRIDGE_INFO_INVALID2,
   CARTRIDGE_INFO_EMPTY,
   CARTRIDGE_INFO_EMPTY2,
   CARTRIDGE_INFO_EMPTY3,
   CARTRIDGE_INFO_OK                            // Valid cartridge
} CartridgeInfoType;
```

4 Definitions

4.1 API Return Values

```
#define PTROBOT_OK                              0
#define PTROBOT_INTERNAL                        500
#define PTROBOT_SEQUENCE                        501
#define PTROBOT_INVALID_ROBOT                   502
#define PTROBOT_INVALID_DRIVE                   503
#define PTROBOT_INVALID_BIN                     504
#define PTROBOT_NODRIVES                        505
#define PTROBOT_OPENCLOSE_FAILED                506
#define PTROBOT_OVERFLOW                        507
#define PTROBOT_NO_PRINTER                      508
#define PTROBOT_PRINTFILE_INVALID               509
#define PTROBOT_PRINTAPP_NOT_INSTALLED          510
#define PTROBOT_PRINTFILE_NOT_FOUND             511
#define PTROBOT_PRN_INVALID                     512
#define PTROBOT_UNSUPPORTED_OPTION              513
#define PTROBOT_DIRNOTFOUND                     514
#define PTROBOT_INVALID_LOCATION                515
#define PTROBOT_MULTDRIVES                      516
#define PTROBOT_INVALID_PRINTER_SETTINGS        517
#define PTROBOT_INVALID_DRIVE_POSITION          518
#define PTROBOT_INVALID_ACTION                  519
#define PTROBOT_FEATURE_NOT_IMPLEMENTED         520
#define PTROBOT_PRINTAPP_OPEN                   521
#define PTROBOT_MISSING_DLL                     522
#define PTROBOT_DRIVE_NOT_READY                 523
#define PTROBOT_INVALID_MEDIA                   524
#define PTROBOT_NO_MEDIA                        525
#define PTROBOT_INVALID_LANG                    526
#define PTROBOT_INVALID_ERROR                   527
#define PTROBOT_BUSY                            528
#define PTROBOT_INVALID_EXTENSION               529
```

4.2 System Errors

```
#define SYSERR_NONE                             0
#define SYSERR_PTR_TRAY                         1
#define SYSERR_CART_CODE                        2
#define SYSERR_INPUT_EMPTY                      3
#define SYSERR_PTR_COMM                         4
#define SYSERR_CLR_EMPTY                        5
#define SYSERR_BLK_EMPTY                        6
#define SYSERR_BOTH_EMPTY                       7
#define SYSERR_PICK                             8
#define SYSERR_ARM_MOVE                         9
```

```
#define SYSERR_CART_MOVE                              10
#define SYSERR_INTERNAL_SW                            12
#define SYSERR_NO_ROBODRIVES                          13
#define SYSERR_OFFLINE                                14
#define SYSERR_COVER_OPEN                             15
#define SYSERR_PRINTER_PICK                           16
#define SYSERR_MULTIPLE_PICK                          17
#define SYSERR_MULTIPLEDISCS_IN_PRINTER               18
#define SYSERR_MULTIPLEDISCS_IN_RECORDER              19
#define SYSERR_DROPPED_DISC_RECORDER                  20
#define SYSERR_DROPPED_DISC_BIN1                      28
#define SYSERR_DROPPED_DISC_BIN2                      29
#define SYSERR_DROPPED_DISC_PRINTER                   33
#define SYSERR_DROPPED_DISC_REJECT                    34
#define SYSERR_DROPPED_DISC_UNKNOWN                   35
#define SYSERR_ALIGNNEEDED                            36
#define SYSERR_COLOR_INVALID                          37
#define SYSERR_BLACK_INVALID                          38
#define SYSERR_BOTH_INVALID                           39
#define SYSERR_NOCARTS                                40
#define SYSERR_K_IN_CMY                               41
#define SYSERR_CMY_IN_K                               42
#define SYSERR_SWAPPED                                43
#define SYSERR_PIGONPRO                               44
#define SYSERR_ALIGNFAILED                            45
#define SYSERR_DROPPED_DISC_PRINTER_FATAL             46
#define SYSERR_MULTIPLEDISCS_IN_RIGHTBIN              47
#define SYSERR_MULTIPLEDISCS_IN_LEFTBIN               48
#define SYSERR_CLR_EMPTY_FINAL                        49
#define SYSERR_BLK_EMPTY_FINAL                        50
#define SYSERR_BOTH_EMPTY_FINAL                       51
#define SYSERR_WAITING_FOR_PRINTER                    52
#define SYSERR_NO_DISC_IN_PRINTER                     53
#define SYSERR_BUSY                                   54
#define SYSERR_PURGE                                  55
#define SYSERR_DOCK_SENSOR                            56
#define SYSERR_ALREADY_PRINTED                        57
#define SYSERR_UNKNOWN_HARDWARE                       58
```

4.3 System State

```
#define SYSSTATE_IDLE        0
#define SYSSTATE_BUSY        1
#define SYSSTATE_ERROR       2
```

4.4 Robot Type

```
#define ROBOT_DISCPUBLISHER              0           // Disc Publisher I
#define ROBOT_DISCPUBLISHERII            1           // Disc Publisher II
#define ROBOT_DISCPUBLISHERPRO           2           // Disc Publisher PRO
#define ROBOT_COMPOSERMAX                3           // ComposerMAX
#define ROBOT_RACKMOUNT_DPII             4           // Disc Publisher XR
#define ROBOT_DISCPUBLISHER_XRP          5           // Disc Publisher XRP
#define ROBOT_DISCPUBLISHER_SE           6           // Disc Publisher SE
```

```
#define ROBOT_DISCPUBLISHERPRO_XI        7           // Disc Publisher Xi Series
#define ROBOT_DISCPUBLISHER_4100         8           // Disc Publisher 4100 Series
#define ROBOT_DISCPUBLISHER_4100_XRP     9           // Disc Publisher 4100 Series (XRP)
#define ROBOT_DISCPUBLISHER_4051         10          // Disc Publisher 4100 Series
#define ROBOT_DISCPUBLISHER_SE3          11          // Disc Publisher SE-3
#define ROBOT_DISCPUBLISHER_4200         12          // Disc Publisher 4200 Series
#define ROBOT_DISCPUBLISHER_4200_XRP     13          // Disc Publisher 4200 Series (XRP
#define ROBOT_DISCPUBLISHER_4052         14          // Disc Publisher 4200 Series
```

4.5 Bin Auto Use

```
#define BIN_INPUT                  0
#define BIN_OUTPUT                 1
```

4.6 Robot Options

```
#define PTOPT_KIOSKMODE            0x00000001
```

4.7 Robot Actions

```
#define PTACT_ALIGNPRINTER         0x00000001
#define PTACT_IGNOREINKLOW         0x00000002
#define PTACT_DISABLEPWRBUTTON     0x00000004
#define PTACT_REINIT_DRIVES        0x00000008
#define PTACT_IDENTIFY             0x00000010
#define PTACT_CANCELCMD            0x00000020
#define PTACT_ENABLEPWRBUTTON      0x00000040
#define PTACT_RESETSYSTEM          0x00000080
#define PTACT_CHECKDISCS           0x00000100  // Check number of discs in bins
#define PTACT_CLEANCARTRIDGES      0x00000200  // Clean the cartridges
#define PTACT_CALIBRATE_ONE_DISC   0x00000400  // SE, II, Pro:  Calibrate for one disc
                                               // (user must put one disc in each bin).
#define PTACT_CHANGE_CARTRIDGE     0x00000800  // SE, II, Pro:  Start the cartridge
                                               // change procedure
#define PTACT_END_CARTRIDGE_CHANGE 0x00001000  // SE: End the cartridge change (can
                                               // close lid also)
#define PTACT_SHIP_POSITION        0x00002000  // SE, II, Pro:  Move the picker to the
                                               // shipping position
#define PTACT_RESET_LEFT_INK_LEVELS 0x00004000 // II:  Clears the ink spits for the LEFT
                                               // cartridge
#define PTACT_RESET_RIGHT_INK_LEVELS 0x00008000 // II:  Clears the ink spits for the
                                               // RIGHT cartridge
#define PTACT_ALLOW_NO_CARTRIDGES  0x00010000  // SE, II, Pro: Allows unit to operate
                                               non-printing robotics without a cartridge
#define PTACT_XI_LIGHT_OFF         0x00020000
#define PTACT_XI_LIGHT_ON          0x00040000
#define PTACT_XI_LIGHT_FLASH       0x00080000
#define PTACT_UNHOOK_PICKER        0x00100000
#define PTACT_AUTOPRINTER_MODE     0x00200000  // DP4100: can perform a faster multiple
                                               copy print-only job by calling
                                               PTRobot_SetPrintCopies() prior to calling
                                               the print function (e.g.
                                                   PTRobot_PrintFile()).
#define     PTACT_FAN_ON            0x00400000  // DP4100: turn on system fan
#define     PTACT_FAN_OFF           0x00800000  // DP4100: turn off system fan
```

4.8 Print Quality

```
#define PQ_LOW          0
#define PQ_MED          1
#define PQ_BETTER       2
#define PQ_HIGH         3
#define PQ_BEST         4
```

4.9 Drive Open Close

```
#define DRIVE_OPEN            0
#define DRIVE_CLOSE           1
```

4.10 Locations

```
#define LOCATION_AUTO         0
#define LOCATION_PRINTER      100
#define LOCATION_REJECT       200
```

4.11 Bus Type

```
#define BUSTYPE_USB           0
#define BUSTYPE_1394          1
```

4.12 Clear Drive

```
#define CLEARDRIVE_NO         0
#define CLEARDRIVE_YES        1
```

4.13 Languages

```
#define ENGLISH         0
#define JAPANESE        1
#define GERMAN          2
#define FRENCH          3
#define SPANISH         4
#define ITALIAN         5
#define CHINESE         6     // Simplified
#define KOREAN          7
#define POLISH          8
#define CHINESE_TRAD    9     // Traditional
```

```
#define CZECH            10
```

4.14 Printer Tray Status

```
#define PRINT_TRAY_IN_WITH_DISC      'D'
#define PRINT_TRAY_IN_NO_DISC        'I'
#define PRINT_TRAY_OUT               'O'
```

4.15 Disc Pick Switch Status

```
#define DISC_PICKER_NO_DISC          'X'
#define DISC_PICKER_HAS_DISC         'O'
```

4.16 Cartridge Types

```
#define CARTRIDGE_NONE               0
#define CARTRIDGE_COLOR              1
#define CARTRIDGE_BLACK              2
#define CARTRIDGE_YELLOW             3
#define CARTRIDGE_CYAN               4
#define CARTRIDGE_MAGENTA            5
#define CARTRIDGE_COLORLOTUS         6
```

4.17 Cartridges Installed

```
#define CARTRIDGE_INSTALLED_NONE     0
#define CARTRIDGE_INSTALLED_COLOR    1
#define CARTRIDGE_INSTALLED_BLACK                2
#define CARTRIDGE_INSTALLED_COLOR_AND_BLACK      3
#define CARTRIDGE_INSTALLED_COLOR_AND_COLOR      4
```

5 Recommended System Error Strings

Most applications will use PTRobot_GetErrorString to display system error messages. However, if you want to use your own error strings instead, below are some suggested error strings for various system errors. Some errors strings will vary depending on the robot type, and not all errors are reported from all robot types. You can determine what robot is connected from `dwRobotType` in `PTRobotInfo` structure (section 4.4 defines the types)

5.1 SYSERR_PTR_TRAY

DiscPublisherI/II:
"Tray movement error. Press the left button on the unit to try again."

DiscPublisher XR/XRP:
"Tray movement error. Open and close the cover to try again."

5.2 SYSERR_CART_CODE

DiscPublisherI/II/ DiscPublisher XR/XRP:
"There was a problem finding the ink cartridges. Open the cover and press the left button. Make sure the color cartridge is installed on the left and the black is on the right. Then close the cover."

5.3 SYSERR_INPUT_EMPTY

DiscPublisherI/II/PRO:
"The input bin is empty. Open the cover and add more discs. Then close the cover and push the left button on the unit."

DiscPublisher XR/XRP:
"The input bin is empty. Open the cover, add more discs, and close the cover to continue."

5.4 SYSERR_PTR_COMM

DiscPublisherI/II/PRO:
"There was an internal printer communications error. Press the left button on the unit to try again."

DiscPublisher XR/XRP:
"There was an internal printer communications error. Open and close the cover to try again."

5.5 SYSERR_CLR_EMPTY

DiscPublisherI/II/PRO:
"WARNING: The color cartridge is LOW on ink. To replace the cartridge, open the cover on the unit and press the left button. Then install the new cartridge and close the cover. To ignore the warning, press the left button."

DiscPublisher XR/XRP:
"WARNING: The color cartridge is LOW on ink. To replace the cartridge, open the cover on the unit and press the left button. Then install the new cartridge and close the cover. To ignore the warning, open and close the cover."

5.6 SYSERR_BLK_EMPTY

DiscPublisherI/II/PRO:
"WARNING: The black cartridge is LOW on ink. To replace the cartridge, open the cover on the unit and press the left button. Then install the new cartridge and close the cover. To ignore the warning, press the left button."

DiscPublisher XR/XRP:
"WARNING: The black cartridge is LOW on ink. To replace the cartridge, open the cover on the unit and press the left button. Then install the new cartridge and close the cover. To ignore the warning, open and close the cover."

5.7 SYSERR_BOTH_EMPTY

DiscPublisherI/II/PRO:
"WARNING: Both ink cartridges are LOW on ink. To replace the cartridges, open the cover on the unit and press the left button. Then install the new cartridges and close the cover. To ignore the warning, press the left button."

DiscPublisher XR/XRP:
"WARNING: Both ink cartridges are LOW on ink. To replace the cartridge, open the cover on the unit and press the left button. Then install the new cartridges and close the cover. To ignore the warning, open and close the cover."

5.8 SYSERR_PICK

DiscPublisherI/II/PRO:
"The disc was not picked. Press the left button on the unit to try again."

DiscPublisher XR/XRP:
"The disc was not picked. Open and close the cover to try again."

5.9 SYSERR_ARM_MOVE

DiscPublisher I:
"There was an arm movement error. Press the left button on the unit to try again."

5.10 SYSERR_CART_MOVE

DiscPublisherI/II/PRO:
"Arm picker error. Press the left button on the unit to try again."

DiscPublisher XR/XRP:
"Arm picker error. Open and close the cover to try again."

5.11 SYSERR_INTERNAL_SW

"There was an internal software error. Please re-start the software."

5.12 SYSERR_NO_ROBODRIVES

"No external recorder drives were found. Re-power the computer and unit, and then re-start the software."

5.13 SYSERR_OFFLINE

"The unit is offline. Please ensure the unit is connected and powered on. You may need to shut down and restart the software."

5.14 SYSERR_COVER_OPEN

"The unit's cover is open. Please close the cover."

5.15 SYSERR_PRINTER_PICK

DiscPublisherI/II/PRO:
"The disc was not picked from the printer. Press the left button to retry."

DiscPublisher XR/XRP:
"The disc was not picked from the printer. Open and close the cover to try again."

5.16 SYSERR_MULTIPLE_PICK

DiscPublisherII/PRO:
"Multiple discs were picked up and moved. Please manually remove any extra discs that were moved, keeping a single disc in place. Then close the cover and press the left button."

DiscPublisher XR/XRP:
"Multiple discs were picked up and moved. Please open the cover and manually remove any extra discs that were moved, keeping a single disc in place. Then close the cover to continue."

5.17 SYSERR_MULTIPLEDISCS_IN_PRINTER

DiscPublisherII/PRO:
"Multiple discs were placed in the printer. Please manually remove any extra discs from the printer, keeping a single disc in place. Then close the cover and press the left button."

DiscPublisher XR/XRP:
"Multiple discs were placed in the printer. Please open the cover and manually remove any extra discs from the printer, keeping a single disc in place. Then close the cover to continue."

5.18 SYSERR_MULTIPLEDISCS_IN_RECORDER

DiscPublisherII/PRO:
"Multiple discs were placed in the recorder. Please manually remove any extra discs from the recorder, keeping a single disc in place. Then close the cover and press the left button."

DiscPublisher XR/XRP:
"Multiple discs were placed in the recorder. Please open the cover and manually remove any extra discs from the printer, keeping a single disc in place. Then close the cover to continue."

5.19 SYSERR_DROPPED_DISC_RECORDER

DiscPublisherII/PRO:
"The disc was dropped while moving into the recorder. Please manually place the disc into the recorder tray. Then close the cover and press the left button."

DiscPublisher XR/XRP:
"The disc was dropped while moving into the recorder. Please open the cover and manually place the disc into the recorder tray. Then close the cover to continue."

5.20 SYSERR_DROPPED_DISC_BIN1

DiscPublisherII/PRO:
"The disc was dropped while moving into the right bin. Please manually place the disc into the right bin. Then close the cover and press the left button."

DiscPublisher XR/XRP:
"The disc was dropped while moving into the right bin. Please open the cover and manually place the disc into the right bin. Then close the cover to continue."

5.21 SYSERR_DROPPED_DISC_BIN2

DiscPublisherII/PRO:
"The disc was dropped while moving into the left bin. Please manually place

the disc into the left bin. Then close the cover and press the left button.”

DiscPublisher XR/XRP:
“The disc was dropped while moving into the left bin. Please open the cover and manually place the disc into the left bin. Then close the cover to continue.”

5.22 SYSERR_DROPPED_DISC_PRINTER

DiscPublisherII/PRO:
“The disc was dropped while moving into the printer. Please manually place the disc into the printer tray. Then close the cover and press the left button.”

DiscPublisher XR/XRP:
“The disc was dropped while moving into the printer. Please open the cover and manually place the disc into the printer tray. Then close the cover to continue.”

5.23 SYSERR_DROPPED_DISC_REJECT

DiscPublisherII/PRO:
“The disc was dropped while moving to the reject area. Please remove the dropped disc. Then close the cover and press the left button.”

DiscPublisher XR/XRP:
“The disc was dropped while moving to the reject area. Please open the cover and remove the dropped disc. Then close the cover to continue.”

5.24 SYSERR_DROPPED_DISC_UNKNOWN

DiscPublisherII/PRO:
“The disc was dropped. Please remove the dropped disc. Then close the cover and press the left button.”

DiscPublisher XR/XRP:
“The disc was dropped. Please open the cover and remove the dropped disc. Then close the cover to continue.”

5.25 SYSERR_ALIGNNEEDED

DiscPublisherPRO:
“The printer cartridges need to be aligned.”

NOTE: Your application can require the user to go to the Printing Preferences in the Printers and Faxes folder to perform this function. Or, you can use the PTRobot_SystemAction call to help the user perform an alignment.

5.26 SYSERR_COLOR_INVALID

DiscPublisherPRO:
"The color cartridge is invalid. Open the cover and press the left button. Change the cartridge and close the cover."

5.27 SYSERR_BLACK_INVALID

DiscPublisherPRO:
"The black cartridge is invalid. Open the cover and press the left button. Change the cartridge and close the cover."

5.28 SYSERR_BOTH_INVALID

DiscPublisherPRO:
"Both cartridges are invalid. Open the cover and press the left button. Change the cartridges and close the cover."

5.29 SYSERR_NOCARTS

DiscPublisherPRO:
"No cartridges are installed. Open the cover and press the left button. Install the cartridges and close the cover."

5.30 SYSERR_K_IN_CMY

DiscPublisherPRO:
"The black cartridge is installed in the color position. Open the cover and press the left button. Change the cartridge and close the cover."

5.31 SYSERR_CMY_IN_K

DiscPublisherPRO:
"The color cartridge is installed in the black position. Open the cover and press the left button. Change the cartridge and close the cover."

5.32 SYSERR_SWAPPED

DiscPublisherPRO:
"The black and color cartridges are swapped. Open the cover and press the left button. Swap the cartridges and close the cover."

5.33 SYSERR_PIGONPRO

DiscPublisherPRO:
"This printer is not compatible with a pigment-based black cartridge. Open the cover and press the left button. Install a dye-based black cartridge and close the cover."

5.34 SYSERR_ALIGNFAILED

DiscPublisherPRO:
"The alignment print failed."

NOTE: Your application can either require the user to go to the Printing Preferences in the Printers and Faxes folder to re-do this function. Or, you can use the PTRobot_SystemAction call to help the user perform another alignment.

5.35 SYSERR_DROPPED_DISC_PRINTER_FATAL

DiscPublisherII/PRO:
"The disc was dropped while moving to/from the printer. Please open the cover and manually remove and discard the disc. Then place a new disc in the recorder, close the cover and press the left button."

DiscPublisher XR/XRP:
"The disc was dropped while moving to/from the printer. Please open the cover and manually remove and discard the disc. Then place a new disc in the recorder and close the cover to continue."

5.36 SYSERR_MULTIPLEDISCS_IN_RIGHTBIN

DiscPublisherII/PRO:
"Multiple discs were placed in the right bin. Please manually move any extra discs to the left bin, keeping a single disc in place. Then close the cover and press the left button."

DiscPublisher XR/XRP:
"Multiple discs were placed in the right bin. Please open the cover and manually move any extra discs to the left bin, keeping a single disc in place. Then close the cover to continue."

5.37 SYSERR_MULTIPLEDISCS_IN_LEFTBIN

DiscPublisherII/PRO:
"Multiple discs were placed in the left bin. Please manually move any extra discs to the right bin, keeping a single disc in place. Then close the cover and press the left button."

DiscPublisher XR/XRP:
"Multiple discs were placed in the left bin. Please open the cover and manually move any extra discs to the right bin, keeping a single disc in place. Then close the cover to continue."

5.38 SYSERR_CLR_EMPTY_FINAL

DiscPublisherPRO:

"WARNING: The color cartridge is Empty. To replace the cartridge, open the cover on the unit and press the left button. Then install the new cartridge and close the cover. To ignore the warning, press the left button."

5.39 SYSERR_BLK_EMPTY_FINAL

DiscPublisherPRO:
"WARNING: The black cartridge is Empty. To replace the cartridge, open the cover on the unit and press the left button. Then install the new cartridge and close the cover. To ignore the warning, press the left button."

5.40 SYSERR_BOTH_EMPTY_FINAL

DiscPublisherPRO:
"WARNING: Both cartridges are Empty. To replace the cartridge, open the cover on the unit and press the left button. Then install the new cartridge and close the cover. To ignore the warning, press the left button."

5.41 SYSERR_WAITING_FOR_PRINTER

"The system timed out waiting for the printer to finish. The disc may not have been printed on."

5.42 SYSERR_NO_DISC_IN_PRINTER

DiscPublisher II, Pro XRP, 4100 XRP:

"Please place the disc back into the printer tray and close the tray. NOTE: First press the button on the right; then pressing the left button will open/close the printer tray."

DiscPublisher SE, Pro Pro Xi, 4100:

"The disc to be printed is missing from the printer tray. Please place the disc back into the printer tray and close the tray. NOTE: You can open/close the printer tray by pressing the left button with the cover closed."

5.43 SYSERR_BUSY

"The system is busy."

5.44 SYSERR_PURGE

DiscPublisher 4100, XRP 4100:

"The system had a problem purging. Restart the system to try again."

5.45 SYSERR_DOCK_SENSOR

DiscPublisher 4100, XRP 4100:

“The system had a problem with the picker connection. Open and close the cover to try again.”

5.46 SYSERR_ALREADY_PRINTER

DiscPublisher 4100, XRP 4100:

“A disc was left in the printer. Remove the disc, close the cover, and press the left button to try again.”

5.47 SYSERR_UNKNOWN_HARDWARE

“The system encountered a hardware error. Restart the system to try again.”

6 Revision History

3/31/11 – document version 2.5

- Added recommended system error strings in sections 5.42 to 5.47.
- Added new 4100 XRP Robot Type. Section 4.4
- Added special case for PTRobot_GetSureThingPreview(). Section 2.4.1

3/8/11 – document version 2.4

- Updated section 2.1.6 Cartridge Types for individual color ink types and added section 2.1.7 for Cartridges Installed. Added several sections in 3. Updated sections 4.2, 4.4, and 4.7. Added few missing API calls.

11/15/07 – document version 2.3

- Added a note about Unicode vs. MCBS versions. (Section 3)

10/26/07 – document version 2.2

- Added new API call: PTRobot_PrintFileWithMerge2(). (Section 2.3.14)
- Added new API call: PTRobot_AddMergeFields(). (Section 2.4.2)
- Added new API call: PTRobot_ClearMergeList(). (Section 2.4.3)
- Added several System Actions (Section 4.7).

7/27/07 – document version 2.1

- Document the fact that PTRobot now supports Disc Publisher SE
- Added two new API calls: PTRobot_GetSureThingPreview() and PTRobot_GetManufactureInfo() Section 2.2.7 and 2.4.1
- Added new structure PTManufactureInfo (Section 3.8)
- Added some new defines

5/16/06 – document version 2.0

- Document the fact that PTRobot now supports Disc Publisher XRP
- Added new API calls PTRobot_GetRobotInfo2() and PTRobot_GetRobotStatus2 (Sections 2.25 and 2.26)
- Added new structures PTRobotInfo2 and PTRobotStatus2 (Sections 3.6 and 3.7)
- Added new defines (Section 4.14 to 4.16)

10/14/05 – document version 1.9

- Document the fact that PTRobot now supports Disc Publisher XR
  Note: ROBOT_RACKMOUNT_DPII is for the Disc Publisher XR

9/14/05 – document version 1.8

- Added new System Errors 46-52 (Section 4.2).
- Added new string descriptions for the newly added system errors (Section 5.35 to 5.41).

7/13/05 – document version 1.7

- Added hRobot parameter to PTRobot_GetErrorString (Section 2.1.11)
- Fixed documentation error for robotic functions (Section 2.3) where the reject position was given as 100 instead of 200.
- Added PTACT_CANCELCMD (Section 4.7)
- Added members to PTRobotStatus structure (Section 3.3)

6/18/05 – document version 1.6

- Updated PTRobot API return values (Section 4.1)
- Updated PTRobot_GetErrorString to also return PTRobot API errors (Section 2.1.11)

6/17/05 - document version 1.5

- Added tszMediaType to the PTMediaInfo structure (Section 3.5)
- Updated recommended system error strings (Section 5)

6/14/05 - document version 1.4

- Added new PTRobot return values (Section 4.1)
- Added PTRobot_SetApplicationID (Section 2.1.12)
- Added PTRobot_GetMediaInfo (Section 2.2.4)
- Added PTRobot_GetErrorString (Section 2.1.11)
- Removed “Cmd Completion Flags” (Previously Section 4.8)
- Removed dwCommandComplete member of PTRobotStatus. (Section 3.3)
- Added PTMediaInfo structure (Section 3.5)
- Added Language definitions (Section 4.13)

6/3/05 - document version 1.3

- Updated notes in PTRobot_EnumRobots

5/23/05 - document version 1.2

- Added links within the document.

5/20/05 - document version 1.1

- Added PrintFileWithMerge() – section 2.3.8
- Added sections 5 “Recommended System Error Strings.”
- Changed BYTE bDriveBusType to DWORD dwDriveBusType in PTRobotInfo structure – section 3.2
- Added ROBOT_RACKMOUNT_DPII Robot Type for the RackMount Disc Publisher II – section 4.4
- Changed all char to TCHAR
- Added PTROBOT_PRINTAPP_OPEN return value