AMP TEK | Mini-X API Programming Guide | Mini-X

Mini-X and Mini-X OEM MX API Programming Guide

1 MINI-X API

The Mini-X API is an Application Programming Interface Library for the Amptek Mini-X Miniature X-Ray Tube System. The Mini-X API functions use standard C calling conventions and parameter types. The Mini-X API creates and controls an instance of a (non-visible) Mini-X Controller Application. The Mini-X Controller Application manages USB communications, controller device programming and X-ray tube control so you don't have to.

1.1 USING MINI-X AND MINI-X OEM TUBES

Mini-X 50kV Tubes have serial numbers 01118880 and above. The new Mini-X API supports 40kV and 50kV tubes. All the examples demonstrate testing for a 50kV tube and updating the display to reflect the tube type.

Visual Basic testing for a 50kV tube:

```
If (MiniXSerialNumber >= 1118880) Then 'this is a 50kV Tube
    picIsoCurve(0).Visible = False     'hide the 40kV Isopower curve
    picIsoCurve(1).Visible = True      'show the 50kV Isopower curve
    lblMAX(1).Caption = "50" 'set the maximum high voltage label to 50kV
    MxDeviceType = ReadMinixOemMxDeviceType 'get tube type (50-70kV only)
    UpdateDisplayByDeviceType(MxDeviceType) 'set the display for the tube
Else                                        'this is a 40kV Tube
    MxDeviceType = -1                       'set the MxDeviceType to 40kV
    UpdateDisplayByDeviceType(MxDeviceType) 'set the display for the tube
End If
```

2 WHAT MINI-X API HOST PROGRAMS CAN DO

Mini-X API host programs can:

- Open and close an instance of a (non-visible) Mini-X Controller Application.
- Monitor Mini-X operations and status.
- Send control commands to a Mini-X Controller.
- Set Mini-X voltage and current.

3 SAMPLE MINI-X API HOST PROGRAM EXAMPLES

Mini-X API host program examples are complete applications. The examples are written in Visual Basic 5 (vbMiniX), Visual C++ 7 (vcMiniX), and Visual Basic .NET 2003 (vbNetMiniX). Example code can be cut and paste into custom applications.

4 WARNINGS AND PRECAUTIONS

4.1 RADIATION PRECAUTIONS

- The Mini-X generates X-ray radiation during normal operation.
- X-rays present a safety hazard.
- The Mini-X must be shielded with radiation shielding (not supplied by Amptek).
- The Mini-X has been designed to focus radiation in the designated output direction, however, radiation in other directions is possible and should be addressed with shielding and/or monitoring in the final application.
- Test radiation shielding from all directions.
- The Mini-X should be operated only by qualified personnel.
- The Mini-X must be used in accordance with local and national regulations.
- Use the Brass Safety Plug whenever possible, use supplied collimators otherwise.

4.2 HARDWARE WARNINGS

- The Mini-X tube housing must never exceed 60℃.
- The Mini-X board temperature is NOT a measure of the Mini-X tube housing temperature.
- Ensure proper heat sinking and/or air forced cooling is present.
- Exceeding the tube housing temperature voids the warranty.
- The X-Ray tube is equipped with a Beryllium window.
- Beryllium windows are fragile, never touch the window!
- Beryllium can be harmful or even toxic if misused.
- Protect the Beryllium window from fluids, vapors and corrosion salts.
- Do not scrape or machine the Beryllium window.
- Do not inhale Beryllium particles if a window gets damaged.

4.3 POWER WARNINGS

- High voltages are present in the Mini-X.
- X-ray sources operate at voltage levels which are potentially life threatening.
- Do not directly access the X-ray source.
- Do not alter the Mini-X Miniature X-Ray Tube System case, hardware, or electronics.
- If direct circuit or X-ray source access is necessary, first disable the power supply.
- Do not exceed the Isopower Curve Wattage.
- Exceeding the Isopower Curve Wattage causes damage to the X-ray source.
- Exceeding the Isopower Curve Wattage voids the warranty.

5 MINI-X API FUNCTIONS

The Mini-X API functions are groups into categories. Following is a summary of the functional categories. C language type true (1) and false (0) values are used except where specified.

5.1 MINI-X CONTROLLER APPLICATION

The Mini-X Controller Application is a non-visible modeless dialog based application. The Mini-X Controller Application manages USB communications, controller device programming and X-ray tube control. The Mini-X Controller Application runs in its' own thread.

The Mini-X Controller Application functions open and close an instance of a (non-visible) Mini-X Controller Application. A test function is provided to determine if an instance exists.

5.1.1 OpenMiniX opens an instance of a Mini-X Controller Application.

An instance of a Mini-X Controller Application must be opened before other Mini-X API functions can be called (except for **isMiniXDlg)**. Before other functions are called **isMiniXDlg** should be called to verify the Mini-X Controller Application is present. A Mini-X monitor **mxmStatusInd** value of **mxstMiniXApplicationReady** indicates the application is ready to connect to the Mini-X controller.

Function Prototypes:

VB	Public Declare Sub OpenMiniX Lib "MiniX" ()
C++	void WINAPI OpenMiniX();

5.1.2 isMiniXDlg tests for an instance of a Mini-X Controller Application.

If an instance of a Mini-X Controller Application is open **isMiniXDlg** will return true or false otherwise. Before other functions are called **isMiniXDlg** should be called to verify the Mini-X Controller Application is present.

Function Prototypes:

VB	Public Declare Function isMiniXDlg Lib "MiniX" () As Byte
C++	byte WINAPI isMiniXDlg();

5.1.3 CloseMiniX Closes an instance of a Mini-X Controller Application.

Before exiting a Mini-X API host program the Mini-X Controller Application must be closed. **CloseMiniX** causes the instance of a Mini-X Controller Application to close and causes the thread to terminate.

Function Prototypes:

VB	Public Declare Sub CloseMiniX Lib "MiniX" ()
C++	void WINAPI CloseMiniX();

5.2 Mini-X Status

Mini-X status functions monitor Mini-X operations and status. All the Mini-X data structures and enumerations are related to Mini-X status functions.

5.2.1 Mini-X Status Data Types and Enumerations

The **MiniX_Monitor** data type holds monitored values from **ReadMiniXMonitor**. The **MiniX_Commands** and **MiniX_Status** enumerations decode status values. The **MiniX_Settings** data type holds corrected settings from **ReadMiniXSettings**.

5.2.1.1 MiniX_Monitor Data Type

The **MiniX_Monitor** data type holds monitored values from **ReadMiniXMonitor**. The Mini-X Controller Application has monitors that indicate the status of the Mini-X X-ray tube, the Mini-X hardware and the Mini-X Controller Application.

5.2.1.2 Mini-X X-Ray Tube Status

mxmHighVoltage_kV is the Mini-X tube monitored high voltage value. This value is read from the Mini-X tube and indicates the actual Mini-X high voltage the X-ray source is delivering.

mxmCurrent_uA is the Mini-X tube monitored current value. This value is read from the Mini-X tube and indicates the actual Mini-X current the X-ray source is delivering.

mxmPower_mW is the Mini-X tube power output in milliwatts. This value must not exceed the values indicated by the Mini-X tube Isopower Curve. ⚠Exceeding the Isopower Curve Wattage causes damage to the X-ray source. ⚠Exceeding the Isopower Curve Wattage voids the warranty.

mxmOutOfRange when true indicates Mini-X tube power output is out of range.

mxmHVOn if true, indicates that the High voltage is on and the Mini-X tube is generating X-Rays. False indicates the high voltage is off and the Mini-X tube in not is generating X-Rays.

WARNING: This device produces X-Rays when energized.

5.2.1.3 Mini-X Hardware Status

mxmTemperatureC is the Mini-x controller board temperature in degrees Celsius (℃).

mxmInterLock is the hardware interlock indicator. (0=Open, 1=Restored (Closed)). The hardware interlock prevents the Mini-X from operating while open.

5.2.1.4 Mini-X Controller Application Status

mxmRefreshed if true, indicates that monitor data refresh was successful.

mxmEnabledCmds indicates which command button buttons should be enabled. Use the **MiniX_Commands** enumeration to decode **mxmEnabledCmds** button enables.

mxmStatusInd holds the Mini-X Controller Application status indicator code. Use the *GetMiniXStatusString* helper function to decode the status code into a text string.

mxmReserved is reserved, the value should be 123.456.

5.2.1.5 MiniX_Monitor Data Type Declarations

VB

```
'holds monitored values from ReadMiniXMonitor
Public Type MiniX_Monitor
    mxmHighVoltage_kV As Double 'high voltage monitor
    mxmCurrent_uA As Double     'current monitor
    mxmPower_mW As Double       'power in milliwatts
    mxmTemperatureC As Double   'temperature in degrees C
    mxmRefreshed As Byte        'monitor data refresh ok
    mxmInterLock As Byte        '0=Open,1=Restored(Closed)
    mxmEnabledCmds As Byte      'command button enables
    mxmStatusInd As Byte        'minix status indicator
    mxmOutOfRange As Byte       'wattage value out of range
    mxmHVOn As Byte             'high voltage on indicator
    mxmReserved As Double       'reserved, should be 123.456

End Type
```

C++

```
//holds monitored values from ReadMiniXMonitor
typedef struct _MiniX_Monitor {
    double mxmHighVoltage_kV;   //high voltage monitor
    double mxmCurrent_uA;       //current monitor
    double mxmPower_mW;         //power in milliwatts
    double mxmTemperatureC;     //temperature in degrees C
    byte mxmRefreshed;          //monitor data refresh ok
    byte mxmInterLock;          //0=Open,1=Restored(Closed)
    byte mxmEnabledCmds;        //command button enables
    byte mxmStatusInd;          //minix status indicator
    byte mxmOutOfRange;         //wattage value out of range
    byte mxmHVOn;               //high voltage on indicator
    double mxmReserved;         //reserved, should be 123.456
} MiniX_Monitor, *LP_MiniX_Monitor;
```

5.2.1.6 MiniX_Settings Data Type Declarations

The **MiniX_Settings** data type holds corrected settings from **ReadMiniXSettings**. Mini-X settings requested with **SetMiniXHV** or **SetMiniXCurrent** may have been corrected during setting. This occurs if the value is out of range or is not a whole number value. **ReadMiniXSettings** reads back the actual settings the Mini-X Controller Application will send to the X-ray source.

VB

```
'holds corrected settings from ReadMiniXSettings
Public Type MiniX_Settings
    HighVoltage_kV As Double    'high voltage setting
    Current_uA As Double        'current setting

End Type
```

C++

```
//holds corrected settings from ReadMiniXSettings
typedef struct _MiniX_Settings {
    double HighVoltage_kV;              //high voltage setting
    double Current_uA;                  //current setting
} MiniX_Settings, *LP_MiniX_Settings;
```

5.2.1.7 MiniX_Commands Enumeration

The **MiniX_Commands** enumeration serves two purposes. **MiniX_Commands** are used as command enable masks for **mxmEnabledCmds** and as command codes for **SendMiniXCommand**.

VB

```
'commands that can be sent with SendMiniXCommand
'ALSO
'command enable masks for mxmEnabledCmds
Public Enum MiniX_Commands
    mxcDisabled = 0             'null command
    mxcStartMiniX = 1           'start minix controller
    mxcHVOn = 2                 'turn high voltage on
    mxcHVOff = 4                'turn high voltage off
    mxcSetHVandCurrent = 8      'set high voltage and current
    mxcExit = 16                'exit controller

End Enum
```

C++

```
//commands that can be sent with SendMiniXCommand
//ALSO
//command enable masks for mxmEnabledCmds
enum MiniX_Commands {
    mxcDisabled = 0,         //null command
    mxcStartMiniX = 1,       //start minix controller
    mxcHVOn = 2,             //turn high voltage on
    mxcHVOff = 4,            //turn high voltage off
    mxcSetHVandCurrent = 8, //set high voltage and current
    mxcExit = 16             //exit controller

};
```

5.2.1.8 MiniX_Status Enumeration

The **MiniX_Status** enumeration is used to decode **mxmStatusInd**. **mxmStatusInd** holds the Mini-X Controller Application status indicator code. Use the *GetMiniXStatusString* helper function to decode the status code into a text string.

VB
```
'minix controller status codes
Public Enum MiniX_Status
    mxstNoStatus                    'no status available
    mxstDriversNotLoaded            'drivers were not found, install drivers
    mxstMiniXApplicationReady       'application is ready to connect to minix
    mxstPortCLOSED                  'minix detected, port closed, will attempt connect
    mxstNoDevicesAttached           'minix is not connected or is not powered
    mxstMiniXControllerSelected     'minix has been found
    mxstMiniXControllerReady        'minix connected and ready for first command
    mxstMiniXControllerFailedToOpen 'minix detected, but failed to open
    mxstNoDeviceSelected            'could not select minix device
    mxstRequestedVoltageOutOfRange  'hv was selected out of range,api will set in range
    mxstRequestedCurrentOutOfRange  'uA was selected out of range,api will set in range
    mxstConnectingToMiniX           'api busy attempting to connect to minix
    mxstUpdatingSettings            'api busy updating settings
    mxstMiniXReady                  'ready for next operation

End Enum
```

C++
```
//minix controller status codes
enum MiniX_Status {
    mxstNoStatus,                    //no status available
    mxstDriversNotLoaded,            //drivers were not found, install drivers
    mxstMiniXApplicationReady,       //application is ready to connect to minix
    mxstPortCLOSED,                  //minix detected, port closed, will attempt connect
    mxstNoDevicesAttached,           //minix is not connected or is not powered
    mxstMiniXControllerSelected,     //minix has been found
    mxstMiniXControllerReady,        //minix connected and ready for first command
    mxstMiniXControllerFailedToOpen,//minix detected, but failed to open
    mxstNoDeviceSelected,            //could not select minix device
    mxstRequestedVoltageOutOfRange,  //hv was selected out of range,api will set in range
    mxstRequestedCurrentOutOfRange,  //uA was selected out of range,api will set in range
    mxstConnectingToMiniX,           //api busy attempting to connect to minix
    mxstUpdatingSettings,            //api busy updating settings
    mxstMiniXReady                   //ready for next operation

};
```

5.2.2 ReadMiniXMonitor reads monitored values.

ReadMiniXMonitor requests a copy the latest status data from the Mini-X Controller Application. If successful, **mxmRefreshed** (monitor data refresh ok indicator) is true. If **mxmRefreshed** is false the controller was busy. Retry to get the latest data. The **MiniX_Monitor** data type holds monitored values from **ReadMiniXMonitor**. The Mini-X Controller Application has monitors that indicate the status of the Mini-X X-ray tube, the Mini-X hardware and the Mini-X Controller Application itself.

Function Prototypes:

VB	Public Declare Sub ReadMiniXMonitor Lib "MiniX" (ByRef MiniXMonitor As MiniX_Monitor)
C++	void WINAPI ReadMiniXMonitor(MiniX_Monitor *MiniXMonitor);

5.2.3 ReadMiniXSerialNumber reads the Mini-X serial number.

The Mini-X serial number should match the serial number on the Mini-X Miniature X-Ray Tube System.

Function Prototypes:

VB	Public Declare Function ReadMiniXSerialNumber Lib "MiniX" () As Long
C++	long WINAPI ReadMiniXSerialNumber();

5.2.4 ReadMiniXSettings reads the Actual Requested Values Set.

The **MiniX_Settings** data type holds corrected settings from **ReadMiniXSettings**. Mini-X settings requested with **SetMiniXHV** or **SetMiniXCurrent** may have been corrected during setting. This occurs if the value is out of range or is not a whole number value. **ReadMiniXSettings** reads back the actual settings the Mini-X Controller Application will send to the X-ray source.

Function Prototypes:

VB	Public Declare Sub ReadMiniXSettings Lib "MiniX" (ByRef MiniXSettings As MiniX_Settings)
C++	void WINAPI ReadMiniXSettings(MiniX_Settings *MiniXSettings);

5.2.5 ReadMinixOemMxDeviceType reads the Mini-X device type.

The Mini-X device type indicator is the tube type of the Mini-X Miniature X-Ray Tube System.

Function Prototypes:

VB	Public Declare Function ReadMinixOemMxDeviceType Lib "MiniX" () As Long
C++	long WINAPI ReadMinixOemMxDeviceType();

Mini-X Device Type Indicator (long)	**Mini-X Tube Type**	**Notes**
-1 (This is manually set for 40kV)	Mini-X 40kV 4 Watt	Detect by serial number
0	Mini-X OEM 70kV 10 Watt	MX Controller
1	Mini-X OEM 50kV 4 Watt	MX Controller
2	Mini-X OEM 50kV 10 Watt	MX Controller
3 (Manually set to -1 for 40kV)	Mini-50kV 4 Watt	Also 3 for 40kV 4 Watt

5.3 Controller Commands and Tube Settings

Controller commands and tube settings control the Mini-X Miniature X-Ray Tube System.

5.3.1 Controller Commands Summary

The **MiniX_Commands** enumeration enumerates the commands that can be sent with **SendMiniXCommand**.

5.3.1.1 mxcDisabled null command

mxcDisabled is a null command. This command will cause the Mini-X Controller Application to exit. This value is in the enumeration for masking purposes. This should **NOT** be sent with **SendMiniXCommand.**

5.3.1.2 mxcStartMiniX start Mini-X Controller

mxcStartMiniX starts the Mini-X controller connection process. This command will cause the Mini-X Controller Application to search for the Mini-X Miniature X-Ray Tube System and connect to the controller. A Mini-X monitor **mxmStatusInd** value of **mxstMiniXControllerReady** indicates that the Mini-X Controller Application is connected to Mini-X Miniature X-Ray Tube System and is ready for first command. A Mini-X monitor **mxmStatusInd** value of **mxstNoDevicesAttached** indicates that the Mini-X Miniature X-Ray Tube System is not connected or is not powered.

The **mxcStartMiniX** command should only be sent if **mxcStartMiniX** is enabled in **MiniX_Monitor.mxmEnabledCmds**.

5.3.1.3 mxcHVOn turn high voltage on

mxcHVOn enables the Mini-X Miniature X-Ray Tube System high voltage and sends the high voltage and current settings to the Mini-X X-ray source. Before sending the **mxcHVOn** command, a **mxcStartMiniX** command must have been successfully completed. After the option for this command has been selected but before this command is run, a dialog should be displayed allowing the operator to abort this operation. A Mini-X monitor **mxmHVOn** value of true indicates that the Mini-X Miniature X-Ray Tube System high voltage is on and extreme caution should be taken.

The **mxcHVOn** command should only be sent if **mxcHVOn** is enabled in **MiniX_Monitor.mxmEnabledCmds**.

WARNING: This device produces X-Rays when energized.

5.3.1.4 mxcHVOff turn high voltage off

mxcHVOff disables the Mini-X Miniature X-Ray Tube System high voltage. Before sending the **mxcHVOff** command, a **mxcStartMiniX** command must have been successfully completed. A Mini-X monitor **mxmHVOn** value of false indicates that the Mini-X Miniature X-Ray Tube System high voltage is off.

The **mxcHVOff** command should only be sent if **mxcHVOff** is enabled in **MiniX_Monitor.mxmEnabledCmds**.

5.3.1.5 mxcSetHVandCurrent set high voltage and current

mxcSetHVandCurrent sends the high voltage and current settings to the Mini-X X-ray source. Before sending the **mxcSetHVandCurrent** command, a **mxcStartMiniX** and **mxcHVOn** command must have been successfully completed. **mxmHighVoltage_kV**, **mxmCurrent_uA** and **mxmPower_mW** display the high voltage, current and power the X-ray source is delivering.

The **mxcSetHVandCurrent** command should only be sent if **mxcSetHVandCurrent** is enabled in **MiniX_Monitor.mxmEnabledCmds**.

WARNING: This device produces X-Rays when energized.
CAUTION: Exceeding the Isopower Curve Wattage causes damage to the X-ray source.
CAUTION: Exceeding the Isopower Curve Wattage voids the warranty.

5.3.1.6 mxcExit exit controller

mxcExit causes a shutdown and exit. The Mini-X Controller Application first attempts to disable the Mini-X controller high voltage enable. Next the application attempts to disconnect from the Mini-X controller. Finally, the Mini-X Controller Application exits and the host thread terminates. The **mxcExit** command can always be sent and should never be disabled.

5.3.2 SendMiniXCommand sends control commands to the Mini-X Controller.

SendMiniXCommand sends the following **MiniX_Commands** to the Mini-X Controller:

- **mxcStartMiniX** starts Mini-X Controller.
- **mxcHVOn** turns high voltage on. **WARNING: This device produces X-Rays when energized.**
- **mxcHVOff** turns high voltage off.
- **mxcSetHVandCurrent** sets high voltage and current.
- **mxcExit** exits the Mini-X Controller.

The **MiniX_Commands** enumeration serves two purposes. **MiniX_Commands** are used as command enable masks for **mxmEnabledCmds** and as command codes for **SendMiniXCommand**.

MiniX_Commands sent with **SendMiniXCommand** should only be sent if the command is enabled in **MiniX_Monitor.mxmEnabledCmds**.

Function Prototypes:

VB	Public Declare Sub SendMiniXCommand Lib "MiniX" (ByVal MiniXCommand As Byte)
C++	void WINAPI SendMiniXCommand(byte MiniXCommand);

5.3.3 SetMiniXHV sets a requested high voltage (kV).

SetMiniXHV sends a requested high voltage to the Mini-X Controller Application. Mini-X settings requested with **SetMiniXHV** or **SetMiniXCurrent** may have been corrected during setting. This occurs if the value is out of range or is not a whole number value. **ReadMiniXSettings** reads back the actual settings the Mini-X Controller Application will send to the X-ray source. The **MiniX_Settings** data type holds corrected settings from **ReadMiniXSettings**.

Function Prototypes:

```
VB      Public Declare Sub SetMiniXHV Lib "MiniX" (ByVal HighVoltage_kV As Double)
C++     void WINAPI SetMiniXHV(double HighVoltage_kV);
```

⚠ CAUTION: Do NOT exceed the Isopower Curve Wattage. Exceeding the Isopower Curve Wattage causes damage to the X-ray source. Exceeding the Isopower Curve Wattage voids the warranty.

5.3.4 SetMiniXCurrent sets a requested current (uA).

SetMiniXCurrent sends a requested current to the Mini-X Controller Application. Mini-X settings requested with **SetMiniXHV** or **SetMiniXCurrent** may have been corrected during setting. This occurs if the value is out of range or is not a whole number value. **ReadMiniXSettings** reads back the actual settings the Mini-X Controller Application will send to the X-ray source. The **MiniX_Settings** data type holds corrected settings from **ReadMiniXSettings**.

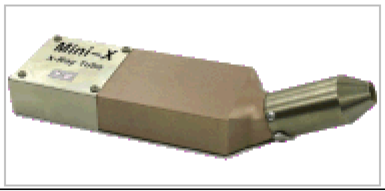
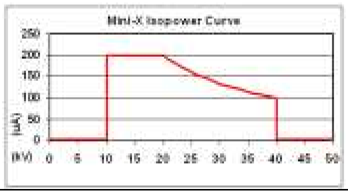
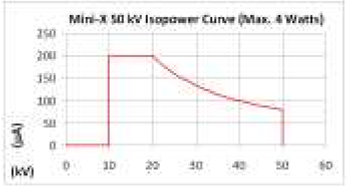
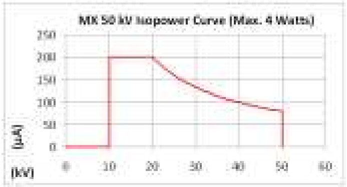
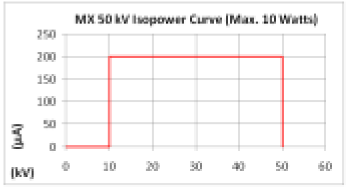
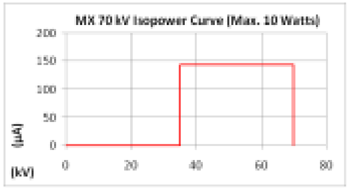
Function Prototypes:

```
VB      Public Declare Sub SetMiniXCurrent Lib "MiniX" (ByVal Current_uA As Double)
C++     void WINAPI SetMiniXCurrent(double Current_uA);
```

⚠ CAUTION: Do NOT exceed the Isopower Curve Wattage. Exceeding the Isopower Curve Wattage causes damage to the X-ray source. Exceeding the Isopower Curve Wattage voids the warranty.

6 ADDITIONAL INFORMATION

6.1 MINI-X TUBES AND CONTROLLERS TABLE

Tube Type	**Controller**	**Mini-X Device Type Indicator (long)**	**Notes**
Mini-X 40kV 4 Watt 	Mini-X 	-1 (This is manually set for 40kV) *API detects 3 for Mini-X NON-OEM Tubes	Detect 40kV by serial number *This Tube is Obsolete
Mini-50kV 4 Watt	Mini-X 	3 (Manually set to -1 for 40kV) *API detects 3 for Mini-X NON-OEM Tubes	API Also returns 3 for 40kV 4 Watt Tube, check for 40kV Tube, set to -1 in local application if 40kV Tube
Mini-X OEM 50kV 4 Watt	MX50 	1	MX Controller
Mini-X OEM 50kV 10 Watt	MX50.10 	2	MX Controller
Mini-X OEM 70kV 10 Watt	MX70 	0	MX Controller

6.2 VISUAL BASIC 5 AND 6 (VB CLASSIC) EXAMPLE UPDATES

Private Sub UpdateDisplayByDeviceType(idxMX As Long)

Updates Mini-X or MX controller setup based on tube type. (See frmMiniX.frm)

Public Sub ReadMiniXSetup40kvMiniX(MiniXConstant As MiniX_Constant)

Public Sub ReadMiniXSetup50kvMiniX (MiniXConstant As MiniX_Constant)

Public Sub ReadMiniXSetup50kv4W_OEM (MiniXConstant As MiniX_Constant)

Public Sub ReadMiniXSetup50kv10W_OEM (MiniXConstant As MiniX_Constant)

Public Sub ReadMiniXSetup70kv10W_OEM (MiniXConstant As MiniX_Constant)

ReadMiniXSetup functions sets controller setup parameters based on the tube type.

Settings are passed using a MiniX_Constant data type.

(See modMxDisplaySettings.bas)

```
'holds Mini-X Fixed Constant values for Display Setup
Public Type MiniX_Constant
    lTubeTypeID As Long                         ' Tube Type Identifier Index
    dblHighVoltageConversionFactor As Double    ' High Voltage Conversion Factor
    dblHighVoltageMin As Double                 ' High Voltage Min
    dblHighVoltageMax As Double                 ' High Voltage Max
    dblDefaultdblHighVoltage As Double          ' Default High Voltage kV
    dblCurrentMin As Double                     ' Current Min
    dblCurrentMax As Double                     ' Current Max
    dblDefaultCurrent As Double                 ' Default Current
    dblWattageMax As Double                     ' Wattage Max
    strTubeType As String                       ' Tube Type Name
    strControllerType As String                 ' Controller Type Name
    strCtrlTypeID As String                     ' Controller Type Short Name

End Type
```

6.3 C++ EXAMPLE UPDATES

```
// The following functions are for display and controller parameter setup
void HideIsoCurveAndMiniX();             // hide values and images until known

// DisplayIsoCurveAndMiniX  display values and images for device type
void DisplayIsoCurveAndMiniX(MINIX_CONTROLLER_ENUM iMX);

bool is40kv;                             // set if device is 40kV Tube
CString MxDblDispFormat(double dblValue); // format function for display values
MiniX_Constant MxDevice;                 // values for display setup
void MXControllerSetup();                // calls controller setup functions

// ReadMXSetup selects setup function by device type
void ReadMXSetup(MINIX_CONTROLLER_ENUM iMX, MiniX_Constant *MiniXConstant); //
MINIX_CONTROLLER_ENUM iMX;               // holds Mini-X device type indicator

//The following functions load a controller parameter setup
void ReadMiniXSetup40kvMiniX(MiniX_Constant *MiniXConstant);
void ReadMiniXSetup50kvMiniX(MiniX_Constant *MiniXConstant);
void ReadMiniXSetup50kv4W_OEM(MiniX_Constant *MiniXConstant);
void ReadMiniXSetup50kv10W_OEM(MiniX_Constant *MiniXConstant);
void ReadMiniXSetup70kv10W_OEM(MiniX_Constant *MiniXConstant);

// ReadMiniXSetupDisplay displays tube setup parameters in dialog
void ReadMiniXSetupDisplay(MiniX_Constant *MiniXConstant);


//holds Mini-X Fixed Constant values for Display Setup
typedef struct _MiniX_Constant {
      long lTubeTypeID;                        // Tube Type Identifier Index
      double dblHighVoltageConversionFactor;  // High Voltage Conversion Factor
      double dblHighVoltageMin;                // High Voltage Min
      double dblHighVoltageMax;                // High Voltage Max
      double dblDefaultdblHighVoltage;         // Default High Voltage kV
      double dblCurrentMin;                    // Current Min
      double dblCurrentMax;                    // Current Max
      double dblDefaultCurrent;                // Default Current
      double dblWattageMax;                    // Wattage Max
      CString strTubeType;                     // Tube Type Name
      CString strControllerType;               // Controller Type Name
      CString strCtrlTypeID;                   // Controller Type Short Name
} MiniX_Constant, *LP_MiniX_Constant;

typedef enum _MINIX_CONTROLLER_ENUM {
      mxceMX70,
      mxceMX50,
      mxceMX5010,
      mxceMiniX,
      mxceUnknownMX,
      mxceUnknownMX1
} MINIX_CONTROLLER_ENUM;
```

6.4 AMPTEK CONTACT INFORMATION

Amptek Inc.
14 Deangelo Drive, Bedford, MA 01730
PH: +1 781 275 2242 FAX: +1 781 275 3470
sales@amptek.com www.amptek.com