Part 3 – API Reference

This section provides a brief description of the API functions and data definitions.

API Data Structures

MTCCreate

```
typedef struct MTCCREATE
{
char   *inputPort;     // name of the input driver
char   *outputPort;    // name of the output driver
char   *mappingFile;  // path and filename of the mapping file
char   *normalFile;    // path and filename of the
normalization file
} MTCCreate;
```

MTCCompType

```
typedef enum
{
       MTCCompNone,
       MTCCompThresh,
       MTCCompDefault
} MTCCompType;
```

MTCQueryType

```
typedef enum
{
       MTCQueryCurrent,
       MTCQueryDefault,
       MTCQueryMax,
       MTCQueryCount
} MTCQueryType;
```

MTCConfig

```
typedef struct MTCCONFIG
{
       int           nTaxels;
       int           nRows;
       int           nCols;
       float         xDimension;
       float         yDimension;
       char         *unitDescriptor;
} MTCConfig;
```

MTCPointer

```
typedef struct MTCPOINTER
{
       BOOL           valid_pointer;
       int            pointer_num;
       float          pressure;
       float          x_pos;
       float          y_pos;
       float          x_vel;
       float          y_vel;
} MTCPointer;
```

API Function Descriptions

Connecting and Disconnecting

MTCHandle MTC_New(MTCCreate mtcCreate)

This function creates an instance of a Tactex MTC object. In the process, this function does the following:

a. Allocates memory for the MTC object.
b. Reads the mapping (configuration) file from disk.
c. Reads the normalization file from disk. If a normalization file with the name specified does not exist, then one is automatically created.
d. Opens the serial communication driver.
e. "pings" the MTC to verify a connection.
f. Determines the MTC Express firmware version.

Parameters:
`mtcCreate` - A structure specifying the input and output serial ports and locations of the mapping and normalization files.

Return:
`MTC_New()` returns a handle to a touch pad instance. It returns `NULL` if the function failed to create a new instance of the MTC. `MTC_GetLastError()` can be called with a NULL argument to determine the cause of failure.

BOOL MTC_Delete(MTCHandle hMTC)

This function stops the communication with the MTC identified by `hMTC` and frees the system resources allocated for the particular MTC.

Parameters:
`hMTC` - handle of the MTC.

Return:
`MTC_Delete()` returns a `TRUE` if successful. Otherwise a `FALSE` is returned.

Version and Configuration

BOOL MTC_GetAPIVersion(char *apiVer)

This function fills the buffer pointed to by `apiVer` with an API version string. The API version sting is null-terminated and consists of no more than 50 characters.

Parameters:
`apiVer` - pointer to a buffer which must be at least 50 bytes long

Return:
`MTC_GetAPIVersion()` returns `TRUE` if successful. Otherwise `FALSE` is returned.

BOOL MTC_GetConfig(MTCHandle hMTC, MTCConfig *config)

This function fills the `config` structure with the configuration data of the specified MTC. If a MTC descriptor string has not been received from the MTC, the host computer will poll the MTC for its descriptor string.

Parameters:
`hMTC` - handle of the MTC from which the configuration data is requested.
`config` - pointer to a `MTCConfig` structure to be filled with the configuration data.

Return:
`MTC_GetVersion()` returns `TRUE` if successful, `FALSE` otherwise.

Int MTC_GetDescriptorString(MTCHandle hMTC,char *descStr, int msecWait)

This function stops the data stream from the MTC, and fills the buffer pointed to by `descStr` with a null-terminated string containing the configuration information obtained from the MTC Express. This function will wait for a response from the MTC for up to `msecWait` milliseconds. The unit descriptor string will also be placed in the internal API memory for future reference when `MTC_GetConfig()` is called. This function is automatically called by the API when MTC_New is called, so a more efficient way to get the descriptor string is to call `MTC_GetConfig`.

Parameters:
`hMTC` - handle of the MTC.
`descStr` - pointer to place unit descriptor string
`msecWait` - Allowed response time specified in milliseconds

Return:
The length of the configuration string is returned. Zero is returned if the MTC failed to respond.

Error Reporting

MTCErr MTC_GetLastError(MTCHandle hMTC)

This function returns the error code associated with the specified MTC. If `hMTC` is `NULL`, then it returns general errors not associated with a specific MTC. If no errors are associated with the specific MTC, then general errors are checked. Once an error has been reported, it is cleared (i.e. subsequent calls to `MTC_GetLastError()` will not report the same event). However, some errors are accumulated and `MTC_GetLastError()` may be called successively in order to retrieve them.

Parameters:
`hMTC` - handle of the MTC.

Return:
The `MTCErr` is returned.

char *MTC_GetErrorString(MTCErr errCode)

This function returns a pointer to a null-terminated string that provides a brief description of the specified error code.

Parameters:
`errCode` - Error code specifying a MTC error return by `MTC_GetLastError()`

Return:
A pointer to the error string specified by `errCode` is returned.

Control of the Data Stream

BOOL MTC_GetIsConnected(MTCHandle hMTC, int msecWait)

This function pings the specified MTC. It returns TRUE if the MTC responds to the ping within the time specified by msecWait.

Parameters:
`hMTC` - handle of the MTC.
`msecWait` – maximum allowable response time specified in milliseconds.

Return:
A `TRUE` is returned if the host computer is connected to the MTC. Otherwise a `FALSE` is returned.

BOOL MTC_StartSendingData(MTCHandle hMTC, MTCCompType compressionType, int msecWait)

This function initiates the stream of data from the MTC. It will wait up to `msecWait` milliseconds for a response before returning. It is necessary to call this function before `MTC_GetNormalizedData` or `MTC_GetRawData`.

Parameters:
`hMTC` - handle of the MTC.
`compressionType` - Instruct the MTC to use a specified compression method
`msecWait` - Allowed response time specified in milliseconds

Return:
A `TRUE` is returned if the data-stream was successfully initiated. If the data-stream was not detected before the time-out period or if the `hMTC` is not valid, a `FALSE` is returned.

BOOL MTC_StopSendingData(MTCHandle hMTC, int msecWait)

This function stops the stream of data from the specified MTC. It will wait up to `msecWait` milliseconds for acknowledgement from the MTC.

Parameters:
`hMTC` - handle of the MTC.
`msecWait` – Maximum allowable response time specified in milliseconds.

Return:
A `TRUE` is returned if the data stream has been successfully initiated. Otherwise a `FALSE` is returned.

float MTC_SetSampleRate(MTCHandle hMTC, float sampleRate, MTCCompType compressionType, int msecWait)

The sample rate of the MTC Express is the rate at which it measures the pressure over the entire pad (i.e. all 72 taxels). Data from the MTC Express is transmitted over the serial cable after each complete sample. Due to bandwidth limitation of the serial connection, higher sampling frequencies must use data compression. This function sets the MTC Express sample rate using the `compressionType` specified. Some MTC Express units may not be capable of sampling at the desired `sampleRate`.

Parameters:
`hMTC` - handle of the MTC.
`sampleRate` - desired MTC sample rate.
`compressionType` - set the MTC compression type.
`msecWait` - Allowed response time specified in milliseconds

Return:
A sample rate closest to the input parameter sampleRate is returned using the `compressionType` specified. A sample rate of *zero* is returned if the operation was unsuccessful.

float MTC_GetSampleRate(MTCHandle hMTC, MTCCompType compressionType, MTCQueryType queryType)

This function returns the rate at which the MTC Express can transmit data using the `compressionType` specified. The `queryType` is used to specify the current setting, default or maximum sample rates. This function does not change the operation of the API or MTC Express, it simply reports what the MTC Express is capable of. Specifying `MTCQueryCurrent` ignores the `compressionType` and retrieves the last sample rate requested of the connected MTC; use `MTC_GetMeasuredSampleRate()` to return the actual sample rate.

Parameters:
`hMTC` - handle of the MTC.
`compressionType` - the MTC compression type.
`queryType` - the current, default or maximum sample rate.

Return:
The (current, default or maximum) sample rate, in Hz, using `compressionType`.

float MTC_GetMeasuredSampleRate(MTCHandle hMTC)

This function returns a measured sample rate from the specified MTC. This sample rate is calculated by dividing the number of samples received from the MTC in a pre-set time period.

Parameters:
`hMTC` - handle of the MTC.

Return:
A measured sample rate, in Hz (samples/second), is returned.

Control of Normalization

BOOL MTC_BeginNormalization(MTCHandle hMTC)

This function starts the automatic normalization (calibration) process. It resets the data calibration parameters and sets a flag internal to the API so that subsequent calls to `MTC_GetRawData` or `MTC_GetNormalizedData` will update those parameters.

Parameters:
`hMTC` - handle of the MTC.

Return:
A `TRUE` is returned if the normalization process is started correctly.

BOOL MTC_EndNormalization(MTCHandle hMTC, BOOL saveToDisk)

This function completes the normalization process. This function freezes the API's internal data calibration parameters. If `saveToDisk` is `TRUE`, the normalization file is overwritten, otherwise the normalization data continues to be used for this session, but the file is not overwritten.

Parameters:
`hMTC` - handle of the MTC.
`saveToDisk` - When `TRUE`, the normalization data is saved to disk.

Return:
`TRUE` is returned if the normalization process is successful.

Data Gathering

long MTC_GetRawData(MTCHandle hMTC, WORD *dataPtr)

This function fills the array pointed to by dataPtr with the taxel raw pressure data received from the MTC Express.

Parameters:
`hMTC` - handle of the MTC.
`dataPtr` - pointer to a memory location capable of storing taxel data. The user is required to ensure that the array is long enough.

Return:
The MTC Express measures, or samples, the pressure on the touch pad up to 200 times per second. After each measurement, the MTC Express transmits the data over the serial port, and tags the data packet with its sample number. `The MTC_GetRawData` function returns that sample number if data is present. Otherwise `0` is returned.

long MTC_GetNormalizedData(MTCHandle hMTC, WORD *dataPtr)

This function fills the array pointed to by dataPtr with normalized taxel pressure data.

Parameters:
`hMTC` - handle of the MTC.
`dataPtr` - pointer to a memory location capable of storing normalized taxel data.

Return:
The sample number is returned if data is present. Otherwise `0` is returned.

Taxel Physical Configuration

float MTC_GetTaxelX(MTCHandle hMTC, int taxelNum)

This function returns the x-coordinate, in mm, of the taxel specified by taxelNum. taxelNum is the index of the array filled by `MTC_GetRawPressure` or `MTC_GetNormalizedPressure`. The value of `taxelNum` must be less than the value in the `nTaxels` field of the `MTCConfig` structure obtained using the `MTC_GetConfig` function.

Parameters:
`hMTC` - handle of the MTC.
`taxelNum` - A specified taxel number.

Return:
This function returns the x-coordinate of the specified taxel in mm.

float MTC_GetTaxelY(MTCHandle hMTC, int taxelNum)

This function returns the y-coordinate, in mm, of the taxel specified by `taxelNum`. `taxelNum` is the index of the array filled by `MTC_GetRawPressure` or `MTC_GetNormalizedPressure`. The value of `taxelNum` must be less than the value in the `nTaxels` field of the `MTCConfig` structure obtained using the `MTC_GetConfig` function.

Parameters:
`hMTC` - handle of the MTC.
`taxelNum` - A specified taxel number.

Return:
This function returns the y-coordinate of the specified taxel in mm.

int MTC_GetTaxelCol (MTCHandle hMTC, int taxelNum)

This function returns the column, between `1` and `nCols`, of the taxel specified by `taxelNum`. `taxelNum` is the index of the array filled by `MTC_GetRawPressure` or `MTC_GetNormalizedPressure`. The value of `taxelNum` must be less than the value in the `nTaxels` field of the `MTCConfig` structure obtained using the `MTC_GetConfig` function.

Parameters:
`hMTC` - handle of the MTC.
`taxelNum` - A specified taxel number.

Return:
This function returns the number of the column in which the specified taxel is located.

int MTC_GetTaxelRow (MTCHandle hMTC, int taxelNum)

This function returns the row, between `1` and `nRows`, of the taxel specified by `taxelNum`. `taxelNum` is the index of the array filled by `MTC_GetRawPressure` or `MTC_GetNormalizedPressure`. The value of `taxelNum` must be less than the value in the `nTaxels` field of the `MTCConfig` structure obtained using the `MTC_GetConfig` function.

Parameters:
`hMTC` - handle of the MTC.
`taxelNum` - A specified taxel number.

Return:
This function returns the number of the row in which the specified taxel is located.

Pointer Calculation

BOOL MTC_InitPointers(MTCHandle hMTC, int num_pointers, MTCPointer *pointer_data_array)

This function initializes the `pointer_data_array` for the specified MTC handle.

Parameters:
`hMTC` - handle of the MTC.
`num_pointers` - the total number of pointers you will ever want to resolve.
`pointer_data_array` - pointer to a `num_pointers` element array of type `MTCPointer`. The user is responsible for allocating the required memory of size `num_pointers*sizeof(MTCPointer).`

Return:
`MTC_InitPointers()` returns a `TRUE` if successful. Otherwise a `FALSE` is returned.

int MTC_ResolvePointers(MTCHandle hMTC, int num_pointers, WORD *pressureData, int threshold, MTCPointer *pointer_data_array, long sampleNum)

This function resolves pointer data from normalized pressure data. The number of resolved pointers is limited to the `num_pointers` parameter. A `threshold` parameter is used to limit the minimum pressure level at which pointers are resolved. This is used to adjust sensitivity.

Parameters:
`hMTC` - handle of the MTC.
`num_pointers` - the total number of pointers you wish to resolve.
`pressureData` - pointer to a memory location containing normalized taxel data.
`threshold` - minimum pressure level at which pointers are resolved.
`pointer_data_array` - pointer to an array of type `MTCPointer` with `num_pointers` elements. This is where all resolved pointer data is stored.
`sampleNum` - the sample number associated with the pressure data.

Return:
The number of pointers resolved is returned. Zero is returned if the MTC handle is invalid or if the pointer algorithm was unable to resolve any pointers.

Appendix A – Technical Specifications

Dimensions	(in, WxDxH)
Package Outline	7.50 x 6.69 x 0.38
Electronics Bay	7.06 x 2.25 x 0.50
Active Tablet Area	5.75 x 3.75
Construction	
Enclosure	Milled Aluminum
Active Area	Polycarbonate
Weight	(oz) 17.0
Minimum Activation Pressure	(psi) 0.4
Maximum Indentation	(in) 0.08
Noise and Vibration Emissions	None
Horizontal pointing accuracy	(in) 0.05
Operating Power Requirements	120 VAC, 60Hz, 8 W
Sampling Rate	(Hz) up to 200
Pressure Resolution	8 bits
Interface RS-232 Serial,	115 KBaud
Connector	DB9 or Mac Serial Adadpter
Storage Temperature	(ºF) -40 to + 120
Operating Temperature	(ºF) +35 to + 100
Relative Humidity	(%) 0 to 95

Note: This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to part 15 of the FCC Rules and ICES 03. These limits are designed to pro-vide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy and, if not in-stalled and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Reorient or relocate the receiving antenna.
- Increase the separation between the equipment and receiver.
- Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.
- Consult the dealer or an experienced radio/TV technician for help.

Caution: Changes or modifications to this equipment, not expressly approved by the manufacturer could void the user's authority to operate the equipment.

Appendix B – Format of the Mapping File

The mapping file has the following format:

```
#       Tactex Controls Inc.
#
#       Touch Tablet Mapping File
$       72      145     95      200     200
#
#       KEY:
#       row     col     x_coord y_coord
#
#LED 1:
        1       1       3.75    90
        4       2       16.25   39
        1       4       41.25   90
        4       5       53.75   39
        1       7       78.75   90
        4       8       91.25   39
        1       10      116.25  90
        4       11      128.75  39
#
#LED 2:
        2       1       3.75    73
        5       2       16.25   22
        2       4       41.25   73
        .
        .
        .
```

All of the lines beginning with # are comments – they have no effect when the mapping file is read by the API. One special line begins with a $ character. There are five numbers in it:

72 – this is the number of taxels in the pad
145 – this is the width of the pad (x-dimension) in mm
95 – this is the height of the pad (y-dimension) in mm
200 – this is the default sample rate in samples-per-second (Hz)
200 – this is the maximum compressed sample rate in Hz

The lines following the header specify the locations of individual taxels. The order of the rows of data is important: the serial stream of data from the MTC Express is sent in the same order as the rows in the mapping file. For each taxel, the following data is given:

Row – the row number
Column – the column number
x-coordinate – the location in mm
y-coordinate – the location in mm

Please refer to section "Pad Configuration (Mapping)" for figures describing the coordinate frames of reference.

Appendix C – Format of the Normalization File

The normalization file must have a specific format. The contents of the normalization file look like this:

```
#             Tactex Controls Inc.
#                Copyright 1998
#          www.tactex.com tci@tactex.com
#
#      Touch Tablet Normalization File
#
#      Min     Scale
       2       9.836538
       1       15.268657
       0       17.049999
       2       12.178572
       0       8.818966
       5       8.119047
       2       6.912162
       .
       .
       .
```

All lines that begin with the # character are remarks and are ignored. Each row of the normalization file corresponds to an individual taxel. The ‘Min’ field indicates the taxel offset (minimum intensity with no pressure applied to pad). The ‘Scale’ field is the scaling factor used to normalize the taxel responses. The normalized response is computed by the formula:

```
NormalizedData = (RawData – Min) * Scale
```

Appendix D – The Fine Print

MTC Express Product Warranty

Tactex Controls Inc. ("Tactex") warrants to the original owner that the MTC Express controller ("MTC Express") delivered in this package will be free from material defects and workmanship for a period of one year following the date of manufacture or the date of purchase as indicated on the returned warranty registration card, which ever is later. This warranty does not extend to damage to the MTC Express incurred during shipping. Such shipping damage should be reported immediately to the claims department of the carrier that delivered the unit. In the event that the MTC Express fails to perform to specification due to a defect in materials or workmanship, customers must contact Tactex to obtain a RMA (Returned Merchandise Authorization) number and return the unit to Tactex Controls Inc. RMA numbers will only be issued to original owners of items who have returned original warranty registration cards and will only be issued where the customer calls for a RMA number before the 1 year warranty term expires. If the item is found to be defective Tactex will replace or repair the product at no charge except for as set forth below, provided that you deliver the item along with the RMA number in the original container and pay the shipping charges. Tactex will not accept the return of any product without an RMA number on the package. Tactex reserves the right to repair or replace the unit at their discretion. Tactex warrants the repaired or replaced unit for 90 days or the remainder of the one year warranty period, which ever is greater. No other warranty is given or implied as to the fitness for use or compatibility for any particular purpose of the MTC Express with any given operating system, software or hardware configuration. Tactex may not be held responsible for damages, loss of profits, personal injury or property damage, expense or inconvenience or any other specific, incidental, speculative or consequential damages caused by the use, mis-use or inability to use the MTC Express Controller and it's bundled Software whether on account of material defect, negligence or otherwise.

The warranty is void if the MTC Express is damaged due to mis-use, abuse, alteration, accident, electrical supply fluctuations or any other environmental occurrence. THERE ARE NO USER SERVICEABLE PARTS INSIDE THE MTC EXPRESS UNIT. Opening the MTC Express unit voids the warranty. This warranty is, and can only be, given by Tactex to the original purchaser and may not be transferred. No warranty is given or implied regarding any third party components or Software that may be bundled with the MTC Express Controller. Notwithstanding the foregoing, Tactex Controls Inc.'s total liability for all claims under this warranty and the agreement of purchase and sale of an MTC Express shall not exceed the price paid for the MTC Express and shall not include any other charges or expenses such as taxes, duties, shipping or insurance charges.

Copyright and Patents

The MTC Express user's guide and any accompanying Software and documentation thereto is subject to copyright 1999/2000 which is owned by Tactex and may not, in whole or in part, be copied, scanned, photocopied or otherwise duplicated in any media digital or otherwise, without express written permission of Tactex. The ownership of all MTC Express Software is retained without exception by Tactex Controls Inc. and is subject to the MTC Express Software license agreement. This product is produced in Canada under license from the Canadian Space Agency US patent 5,917,180. Other patents pending.

Trademarks

"Smart Fabric", "Capturing Touch" and MTC Express are trademarks of Tactex Controls Inc.. Windows and Microsoft are registered trademarks of Microsoft Corporation, MAX and MSP are trademarks of Opcode/IRCAM/Cycling74. All other products mentioned may be trademarks of other companies and are used for illustrative purposes only without intent to infringe.

Y2K

Tactex's MTC Express poses no Year 2000 issue. They are all Year 2000 Ready. While your Tactex product is ready for the Year 2000, other parts of the computer system which support your Tactex MTC Express, such as computer hardware, firmware, operating systems, and software applications, may not be Year 2000 ready. Users need to test their equipment and software, and contact the respective manufacturers of those products for more information. Tactex expressly disclaims any warranty of non-Tactex products used in conjunction with your MTC Express.

WARNING ABOUT USE

The MTC Express is intended for use as a computer input device for the purposes of musical entertainment and graphic arts only. Tactex products are not designed or built with components intended to ensure a level of reliability for applications where the safety of individuals may be concerned. Do NOT use the MTC Express in clinical or medical applications. Do NOT use the MTC Express to control moving vehicles, wheel chairs, industrial equipment, or machinery. Do NOT use the MTC Express to control any equipment whereby the failure of the MTC Express or Software may cause injury to a person or damage to property.

MTC Express Software License Agreement

Your MTC Express package has been delivered with application Software on a CD. Please read this Software license agreement ("license") carefully before installation. By installing the Software, you are agreeing to be bound by the terms of this license.

1. License: The Software accompanying this license, whether on disk, on compact disc, in read only memory, or any other media, and the related documentation (collectively called, the "Tactex Software") are licensed, not sold, to you by Tactex Controls Inc. ("Tactex"). The Tactex Software in this package and any copies, modifications and distributions which this license authorizes you to make are subject to this license.

2. Permitted uses and restrictions The Tactex Software is licensed for use exclusively with the Tactex MTC Express. You do not have license to use all or part of the Tactex Software in any application which does not require the user to own and operate a Tactex MTC Express. Except as expressly permitted in this license, you may not decompile, reverse engineer, disassemble, modify, rent, lease, loan, sublicense, distribute or create derivative works based upon the Tactex Software in whole or part or transmit the Tactex Software over a network or from one computer to another. Your rights under this license will terminate automatically without notice from Tactex if you fail to comply with any term(s) of this license. In addition, Tactex reserves the right to terminate this license if a new version of the operating system is released which is incompatible with the Tactex Software.

3. Disclaimer of warranty: Some of the Tactex Software may be designated as alpha, beta, development, pre-release, untested, or not fully tested versions of the Tactex Software. Such Tactex Software may contain errors that could cause failures or loss of data, and may be incomplete or contain inaccuracies. You expressly acknowledge and agree that use of the Tactex Software is at your sole risk. The Tactex Software is provided "as is" and without warranty of any kind and Tactex and Tactex's licensor(s) (for the purposes of sections 3 and 4, Tactex and Tactex's licensor(s) shall be collectively referred to as "Tactex") expressly disclaim all warranties, express or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Tactex does not warrant that the functions contained in the Tactex Software will meet your requirements, or that the operation of the Tactex Software will be uninterrupted or error-free, or that defects in the Tactex Software will be corrected. Furthermore, Tactex does not warrant or make any representations regarding the use or the results of the use of the Tactex Software or in terms of their correctness, accuracy, reliability, or otherwise. No oral or written information or advice

given by Tactex or a Tactex authorized representative shall create a warranty or in any way increase the scope of this warranty. Should the Tactex Software prove defective, you (and not Tactex or a Tactex authorized representative) assume the entire cost of all necessary servicing, repair or correction. The license fees for the Tactex Software reflect this allocation of risk. Some jurisdictions do not allow the exclusion of implied warranties, so the above exclusion may not apply to you.

4. Limitation of liability: Under no circumstances, shall Tactex be liable for any incidental, special or consequential damages that result from the use or inability to use the Tactex Software. In no event shall Tactex's total liability to you for all damages, losses, and causes of action (whether in contract, tort (including negligence) or otherwise) exceed the price paid for the MTC Express accompanying the Tactex Software.

5. Controlling law and severability: If there is a local subsidiary of Tactex in the country in which the Tactex Software license was obtained, then the local law in which the subsidiary sits shall govern this license. Otherwise, this license shall be governed by the laws of Canada and the Province of British Columbia. If for any reason a court of competent jurisdiction finds any provision, or portion thereof, to be unenforceable, the remainder of this license shall continue in full force and effect.

6. Complete agreement: This license constitutes the entire agreement between the parties with respect to the use of the Tactex Software and supersedes all prior or contemporaneous understandings regarding such subject matter. No amendment to or modification of this license will be binding unless in writing and signed by Tactex.