The 5DT Data Glove Driver

Reference Manual

5DT

Version 1.00

January 2000

Table of Contents

1. Introduction

The 5DT Data Glove Driver provides access to the 5DT range of data gloves at an intermediate level.

The Windows 95/98/NT version is provided in the form of a C/C++ header (.h) file, a Microsoft Visual C++ library (.lib) file and a dynamic link library (.dll) file.

The Linux version is provided in form of a C/C++ header (.h) file, and a dynamic library (.so) file.

The driver functionality includes:

- Multiple instances
- Easy initialization and shutdown
- Basic (raw) sensor values
- Scaled (auto-calibrated) sensor values
- Calibration functions
- Basic gesture recognition
- Cross-Platform Application Programming Interface

2. Using the glove driver

The 5DT Data Glove Driver is easy to implement. Use the following guidelines:

2.1 Windows 95/98/NT

i. Make sure that the header file `fglove.h`, the library file `fglove.lib` and the dynamic link library file `fglove.dll` reside in the current (application) directory, or somewhere that they can be found. The file `fglove.dll` may be copied into your Windows system directory.
ii. Include the header file `fglove.h` in the application where necessary.
iii. Add the library file `fglove.lib` to the link process.

There is also a debug version of the driver (`fgloved.lib`, `fgloved.dll`) which outputs debug messages to the debugger.

2.2 Linux

The driver requires the Linux Posix threads library `libpthread.so`, which is usually installed with Linux. The driver is linked to libc6.

2.2.1 Installing the library

The driver is a dynamic library file (`libfglove.so`) which must be installed somewhere where applications can find it. If you have root access to the system, the easiest method is to copy the file into the `/usr/lib` directory, e.g.:

```
# cp libfglove.so /usr/lib
```

If you do not have root access to the system, you should place the library somewhere in your home directory, and then set the `LD_LIBRARY_PATH` environment variable to include the full path of the directory in which you have placed the library. This will indicate to the dynamic library loader where to find the file. For example:

```
$ mkdir /home/yourhomedir/libs
$ cp libfglove.so /home/yourhomedir/libs
$ export LD_LIBRARY_PATH="/home/yourhomedir/libs"
```

In the above example, replace "yourhomedir" with your own home directory name.

2.2.2 Setting up access to the serial port

The 5DT Data Glove accesses the serial port using the standard Unix/Linux device files in the `/dev` directory. Applications that use the glove should typically give the user the option of specifying which device to check, such as `/dev/ttyS1`. It is recommended that you create a symbolic link `/dev/fglove` to your device file, e.g:

```
# cd /dev
# ln -s ttyS1 fglove
```

This may make the setup of applications easier should you need to change the port that the glove is connected to.

It is also required that the user of the glove has read/write access to the serial port device file. By default, normally only the root user has these rights. If non-root users on the system will be using the glove, the root user must grant access rights to the device file. For example, the following command will give everyone on the system full access to the serial port `/dev/ttyS1`:

```
# chmod 777 /dev/ttyS1
```

2.2.3 The header file

The header file (`fglove.h`) should be copied somewhere where your compiler can find it. If you have root access, the easiest is to copy it to `/usr/include`. If not, then just place the header file in the same directory as your application.

2.3 Sample source code

Sample source code that makes use of the glove driver is distributed with the driver.

3. Supported gloves

The glove driver supports all 5DT Data Gloves. The current version implements 18 possible sensors, and includes the roll and pitch sensors of the original 5DT Data Glove 5. The driver attempts to map values to all sensor outputs. If it is unable to do so the sensor value defaults to a sensible value. This value can be adjusted by forcing a specific value. To the application programmer the driver therefore appears the same regardless of the type of glove that is connected.

3.1 Sensor mappings for the 5DT Data Glove 5

The sensors on the 5DT Data Glove 5 are positioned as in Fig. 1.

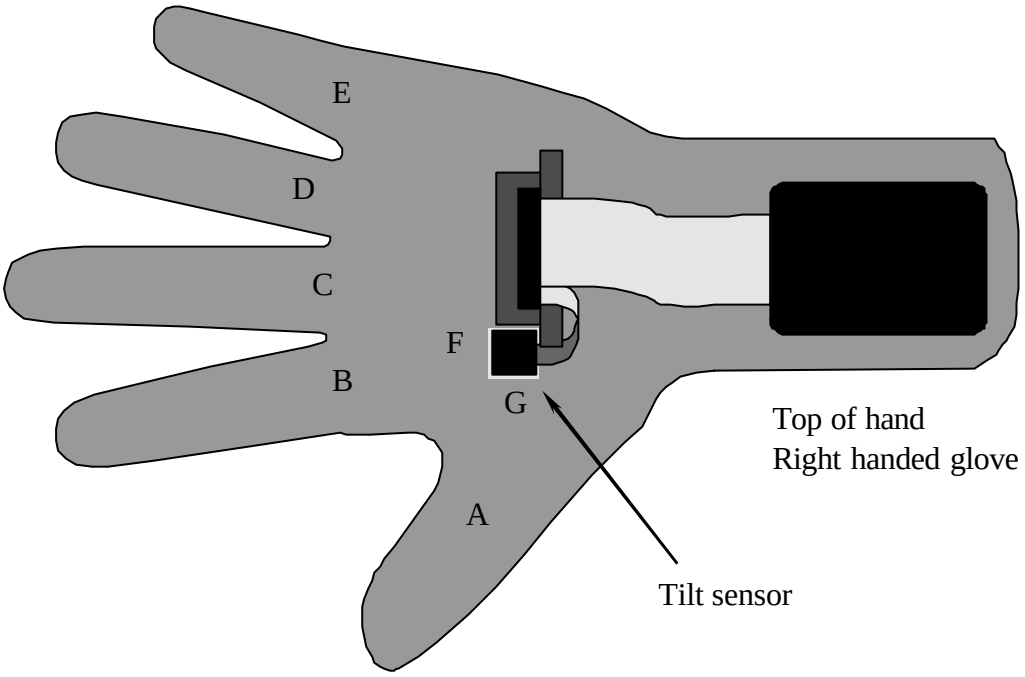

Figure 1 - Sensor positions for the 5DT Data Glove 5

Sensor	*Driver Sensor Index*	*Description*
A	0,1*****	Thumb flexure
B	3,4*****	Index finger flexure
C	6,7*****	Middle finger flexure
D	9,10*****	Ring finger flexure
E	12,13*****	Little finger flexure
F	16	Pitch angle of tilt sensor
G	17	Roll angle of tilt sensor

Table 1 - Sensor mappings for 5DT Data Glove 5

***** Both these driver sensor indices will return the same value when the 5DT Data Glove 5 is used.

3.2 Sensor mappings for the 5DT Data Glove 16

The sensors on the 5DT Data Glove 16 are positioned as in Fig. 2

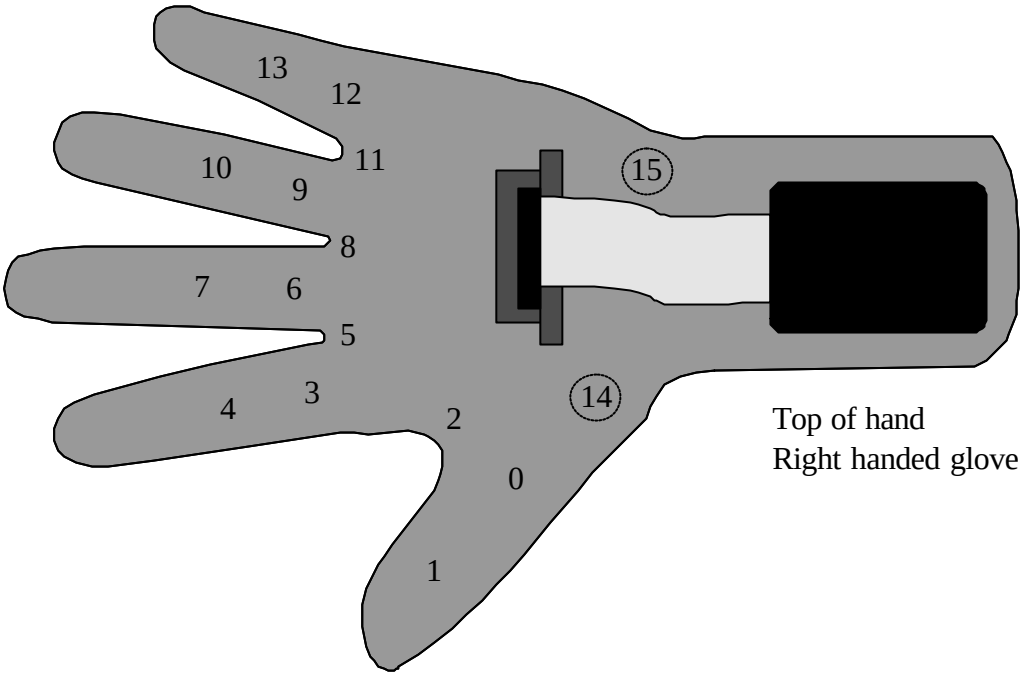

Figure 2 - Sensor mappings for the 5DT Data Glove 16

Sensor	*Driver Sensor Index*	*Description*
0	0	Thumb flexure (lower joint)
1	1	Thumb flexure (second joint)
2	2	Thumb-index finger abduction
3	3	Index finger flexure (at knuckle)
4	4	Index finger flexure (second joint)
5	5	Index-middle finger abduction
6	6	Middle finger flexure (at knuckle)
7	7	Middle finger flexure (second joint)
8	8	Middle-ring finger abduction
9	9	Ring finger flexure (at knuckle)
10	10	Ring finger flexure (second joint)
11	11	Ring-little finger abduction
12	12	Little finger flexure (at knuckle)
13	13	Little finger flexure (second joint)
14	14	Thumb translation [not yet implemented]
15	15	Wrist flexure [not yet implemented]

Table 2 - Sensor mappings for 5DT Data Glove 16

4. Function reference

The following functions are provided by the driver:

`fdGlove *fdOpen(char *pPort)`

Initializes the glove device on the specified port.

Return value
Returns a pointer to the glove device (`fdGlove *`). NULL is returned if an error occurred.

Parameter
`pPort`
Pointer to a zero terminated ASCII string containing the name of the communication port. Valid values on Windows range from "COM1" to "COM8". Unix/Linux port names will differ.

Remarks
Do not attempt to alter the contents of the returned pointer directly, use the functions provided instead.

`int fdClose(fdGlove *pFG)`

Frees the glove device and communications port.

Return value
Returns nonzero if successful, zero if an error occurred.

Parameters
`pFG`
Pointer to a glove device. This is the value returned by `fdOpen()`.

Remarks
It is important to call this function when you are finished using the glove.

`int fdGetGloveHand(fdGlove *pFG)`

Obtains the handedness (left or right handed) of the glove.

Return value
Returns either `FD_HAND_LEFT` or `FD_HAND_RIGHT`, as defined by the enumerated type `EfdGloveHand`.

Parameters
`pFG`
Pointer to a glove device. This is the value returned by `fdOpen()`.

Remarks
None.

`int fdGetGloveType(fdGlove *pFG)`

Obtains the type of the currently connected glove.

Return value
Returns one of `FD_GLOVENONE`, `FD_GLOVE7`, `FD_GLOVE7W`, `FD_GLOVE16` or `FD_GLOVE16W`, as defined by the enumerated type `EfdGloveTypes`.

Parameters
`pFG`
Pointer to a glove device. This is the value returned by `fdOpen()`.

Remarks
`FD_GLOVE7` and `FD_GLOVE7W` refer to the original 5+2 (tilt angles) sensor glove (5DT Data Glove 5). The `W` suffix indicates a wireless model. `FD_GLOVE16` and `FD_GLOVE16W` refer to the 16 sensor glove. In order to accommodate both glove types the `fdGetNumSensors()` function currently returns 18 sensors. The additional two sensors are defined as the original tilt angles that are not present in the 16-sensor glove. See the description of `fdGetNumSensors()` for more details.

`int fdGetNumSensors(fdGlove *pFG)`

Obtains the number of available sensors values the driver can make available.

Return value
Returns the number of sensors. Currently it is fixed at 18, but future driver releases may differ.

Parameters
`pFG`
Pointer to a glove device. This is the value returned by `fdOpen()`.

Remarks
Although the 5-sensor glove can measure only average flexure, the driver will attempt to fill in missing values. The number of sensors returned can therefore be of a higher dimension. The enumerated type `EfdSensors` defines the finger mapping for each sensor.

`void fdGetSensorRawAll(fdGlove *pFG, unsigned short *pData)`

Obtains the most recent raw sensor values from the currently connected glove.

Return value
None.

Parameters
`pFG`
Pointer to a glove device. This is the value returned by `fdOpen()`.

pData
Pointer to an array of 16-bit integers that will contain the raw sensor values. The size of the array must always match the value returned by `fdGetNumSensors()`.

Remarks
Currently the raw sensor samples are all 12 bit unsigned values. The range is therefore from 0 to 4095. Note that this is not the dynamic range of the sensors. There can be severe offset values associated with each sensor. The enumerated type `EfdSensors` defines the finger mapping for each sensor.

`unsigned short fdGetSensorRaw(fdGlove *pFG, int nSensor)`

Obtains the most recent raw sensor value for a specific sensor from the currently connected glove.

Return value
Returns a 16-bit integer. See `fdGetSensorRawAll()` for details.

Parameters
pFG
Pointer to a glove device. This is the value returned by `fdOpen()`.

nSensor
Index of the sensor that is being set. The value must lie in the range given by the enumerated type `EfdSensors`, or alternatively from zero to the value returned by `fdGetNumSensors()` minus one.

Remarks
The enumerated type `EfdSensors` defines the finger mapping for each sensor.

`void fdSetSensorRawAll(fdGlove *pFG, unsigned short *pData)`

Forces the raw value for all the sensors.

Return value
None.

Parameters
pFG
Pointer to a glove device. This is the value returned by `fdOpen()`.

pData
Pointer to an array of 16-bit integers that will contain the raw sensor values. The size of the array must always match the value returned by `fdGetNumSensors()`.

Remarks
Currently the raw sensor samples are all 12 bit unsigned values. The range is therefore from 0 to 4095. The enumerated type `EfdSensors` defines the finger mapping for each sensor. Forcing a sensor value will result in a raw and scaled output other than the default zero. Values that can be mapped will be overwritten, rendering the forced value void.

`void fdSetSensorRaw(fdGlove *pFG, int nSensor, unsigned short nRaw)`

Forces the raw value for a specific sensor.

Return value
None.

Parameters
`pFG`
Pointer to a glove device. This is the value returned by `fdOpen()`.

`nSensor`
Index of the sensor that is being set. The value must lie in the range given by the enumerated type `EfdSensors`, or alternatively from zero to the value returned by `fdGetNumSensors()` minus one.

`nRaw`
16 bit raw value of the sensor. If the sensor is unmapped, the scaling calculations will proceed as normal.

Remarks
The enumerated type `EfdSensors` defines the finger mapping for each sensor. This function is only useful for sensors that cannot be mapped by a specific hardware device. Forcing a sensor value will result in a raw and scaled output other than the default zero. Values that can be mapped will be overwritten, rendering the forced value void.

`void fdGetSensorScaledAll(fdGlove *pFG, float *pData)`

Obtains the most recent scaled (auto-calibrated) sensor values from the currently connected glove.

Return value
None.

Parameters
`pFG`
Pointer to a glove device. This is the value returned by `fdOpen()`.

`pData`
Pointer to an array of floating point numbers that will contain the scaled sensor values. The size of the array must always match the value returned by `fdGetNumSensors()`.

Remarks
The sensor range is a value from zero to the value defined by the `fdSetSensorMax()` and `fdSetSensorMaxAll()` functions. The glove driver defaults to a range of [0...1]. The automatic calibration process is described in section 6. The enumerated type `EfdSensors` defines the finger mapping for each sensor.

`float fdGetSensorScaled(fdGlove *pFG, int nSensor)`

Obtains the most recent scaled (auto-calibrated) value for a specific sensor from the currently connected glove.

Return value
Returns a floating point sensor value.

Parameters
`pFG`
Pointer to a glove device. This is the value returned by `fdOpen()`.

`nSensor`
Index of the sensor that is being set. The value must lie in the range given by the enumerated type `EfdSensors`, or alternatively from zero to the value returned by `fdGetNumSensors()` minus one.

Remarks
The sensor range is a value from zero to the value defined by the `fdSetSensorMax()` and `fdSetSensorMaxAll()` functions. The glove driver defaults to a range of [0...1]. The automatic calibration process is described in section 6. The enumerated type `EfdSensors` defines the finger mapping for each sensor.

`int fdGetNumGestures(fdGlove *pFG)`

Obtains the number of available gestures that can be recognized by the glove driver.

Return value
Returns the number of available gestures. Currently 16 different gestures are supported. Refer to section 5 for details.

Parameters
`pFG`
Pointer to a glove device. This is the value returned by `fdOpen()`.

Remarks
None.

`int fdGetGesture(fdGlove *pFG)`

Obtains the current gesture being performed.

Return value
Returns the current gesture being performed. Refer to section 5 for details.

Parameters
`pFG`
Pointer to a glove device. This is the value returned by `fdOpen()`.

Remarks
None.

---

void fdGetCalibrationAll(fdGlove *pFG, unsigned short *pUpper, unsigned short *pLower)

Obtains the current auto-calibration settings of the driver.

Return value
None.

Parameters
`pFG`
Pointer to a glove device. This is the value returned by `fdOpen()`.

`pUpper` and `pLower`
Arrays of 16 bit unsigned integers that will contain the maximum and minimum raw sensor values. The size of each array must always match the value returned by `fdGetNumSensors()`. Refer to section 6 for details.

Remarks
None.

---

void fdGetCalibration(fdGlove *pFG, int nSensor, unsigned short *pUpper, unsigned short *pLower)

Obtains the current auto-calibration settings of the driver for a specific sensor.

Return value
None.

Parameters
`pFG`
Pointer to a glove device. This is the value returned by `fdOpen()`.

`nSensor`
Index of the sensor that is being set. The value must lie in the range given by the enumerated type `EfdSensors`, or alternatively from zero to the value returned by `fdGetNumSensors()` minus one.

`pUpper` and `pLower`
Pointers to 16 bit unsigned integers containing the maximum and minimum raw sensor values. Refer to section 6 for details.

Remarks
None.

---

void fdSetCalibrationAll(fdGlove *pFG, unsigned short *pUpper, unsigned short *pLower)

Resets the current auto-calibration settings of the driver to user defined values.

Return value
None.

Parameters
`pFG`
Pointer to a glove device. This is the value returned by `fdOpen()`.

`pUpper` and `pLower`
Arrays of 16 bit unsigned integers containing the maximum and minimum raw sensor values. The size of each array must always match the value returned by `fdGetNumSensors()`. Refer to section 6 for details.

Remarks
For unmapped sensors it would be sensible to set the upper and lower calibration settings above and below the raw value forced with `fdSetSensorRaw()`and `fdSetSensorRawAll()`.

`void fdSetCalibration(fdGlove *pFG, int nSensor, unsigned short nUpper, unsigned short nLower)`

Resets the current auto-calibration settings of the driver for a specific sensor to user defined values.

Return value
None.

Parameters
`pFG`
Pointer to a glove device. This is the value returned by `fdOpen()`.

`nSensor`
Index of the sensor that is being set. The value must lie in the range given by the enumerated type `EfdSensors`, or alternatively from zero to the value returned by `fdGetNumSensors()` minus one.

`nUpper` and `nLower`
16 bit unsigned integers containing the maximum and minimum raw sensor values. Refer to section 6 for details.

Remarks
For unmapped sensors it would be sensible to set the upper and lower calibration settings above and below the raw value forced with `fdSetSensorRaw()`and `fdSetSensorRawAll()`.

`void fdResetCalibration(fdGlove *pFG)`

Resets the internal auto-calibration settings of the driver to appropriate default values.

Return value
None.

Parameters
`pFG`
Pointer to a glove device. This is the value returned by `fdOpen()`.

Remarks
This function is similar to the `fdSetCalibrationAll()` function with each of the upper and lower calibration array values set to 0 and 4095 respectively. This function, or any of the other calibration functions, should be called whenever the application starts up or the glove changes users during run-time. For unmapped sensors the upper and lower calibration values are set to 4095 and 0 respectively, which is the *inverse* of the auto-calibration settings.

`void fdGetSensorMaxAll(fdGlove *pFG, float *pMax)`

Obtains the maximum scaled value for each sensor.

Return value
None.

Parameters
`pFG`
Pointer to a glove device. This is the value returned by `fdOpen()`.

`pMax`
Array of floating point values that will contain the maximum scaled sensor values. The size of the array must always match the value returned by `fdGetNumSensors()`.

Remarks
The glove driver defaults to a maximum scaled value of 1 for each sensor.

`float fdGetSensorMax(fdGlove *pFG, int nSensor)`

Obtains the maximum scaled value for a specific sensor.

Return value
Returns the maximum scaled values of the sensor.

Parameters
`pFG`
Pointer to a glove device. This is the value returned by `fdOpen()`.

`nSensor`
Index of the sensor that is being queried. The value must lie in the range given by the enumerated type `EfdSensors`, or alternatively from zero to the value returned by `fdGetNumSensors()` minus one.

Remarks

The glove driver defaults to a maximum scaled value of 1 for each sensor.

`void fdSetSensorMaxAll(fdGlove *pFG, float *pMax)`

Sets the maximum scaled value for each sensor.

Return value
None.

Parameters
`pFG`
Pointer to a glove device. This is the value returned by `fdOpen()`.

`pMax`
Array of floating point values that contains the maximum scaled sensor values. The size of the array must always match the value returned by `fdGetNumSensors()`.

Remarks
The glove driver defaults to a maximum scaled value of 1 for each sensor.

`void fdSetSensorMax(fdGlove *pFG, int nSensor, float fMax)`

Sets the maximum scaled value for a specific sensor.

Return value
None.

Parameters
`pFG`
Pointer to a glove device. This is the value returned by `fdOpen()`.

`nSensor`
Index of the sensor that is being set. The value must lie in the range given by the enumerated type `EfdSensors`, or alternatively from zero to the value returned by `fdGetNumSensors()` minus one.

`fMax`
A floating point value that contains the maximum scaled sensor value.

Remarks
The glove driver defaults to a maximum scaled value of 1 for each sensor.

`void fdGetThresholdAll(fdGlove *pFG, float *pUpper, float *pLower)`

Obtains the current gesture recognition threshold settings of the driver.

Return value
None.

Parameters
`pFG`
Pointer to a glove device. This is the value returned by `fdOpen()`.

`pUpper` and `pLower`
Arrays of floating point numbers that will contain the maximum and minimum threshold values. The size of each array must always match the value returned by `fdGetNumSensors()`. Refer to section 6 for details.

Remarks
None.

`void fdGetThreshold(fdGlove *pFG, int nSensor, float *pUpper, float *pLower)`

Obtains the current gesture recognition threshold settings of the driver for a specific sensor.

Return value
None.

Parameters
`pFG`
Pointer to a glove device. This is the value returned by `fdOpen()`.

`nSensor`
Index of the sensor that is being queried. The value must lie in the range given by the enumerated type `EfdSensors`, or alternatively from zero to the value returned by `fdGetNumSensors()` minus one.

`pUpper` and `pLower`
Pointers to floating point numbers that will contain the maximum and minimum threshold values. Refer to section 6 for details.

Remarks
None.

`void fdSetThresholdAll(fdGlove *pFG, float *pUpper, float *pLower)`

Sets the current gesture recognition threshold settings of the driver.

Return value
None.

Parameters
`pFG`
Pointer to a glove device. This is the value returned by `fdOpen()`.

`pUpper` and `pLower`

Arrays of floating point numbers that contains the maximum and minimum threshold values. The size of each array must always match the value returned by `fdGetNumSensors()`. Refer to section 6 for details.

Remarks
None.

`void fdSetThreshold(fdGlove *pFG, int nSensor, float fUpper, float fLower)`

Sets the current gesture recognition threshold settings of the driver for a specific sensor.

Return value
None.

Parameters
`pFG`
Pointer to a glove device. This is the value returned by `fdOpen()`.

`nSensor`
Index of the sensor that is being set. The value must lie in the range given by the enumerated type `EfdSensors`, or alternatively from zero to the value returned by `fdGetNumSensors()` minus one.

`fUpper` and `fLower`
Floating point numbers that contain the maximum and minimum threshold values. Refer to section 6 for details.

Remarks
None.

`void fdGetGloveInfo(fdGlove *pFG, unsigned char *pData)`

Obtains the information data block of the currently connected glove.

Return value
None.

Parameters
`pFG`
Pointer to a glove device. This is the value returned by `fdOpen()`.

`pData`
Array of 32 bytes that will contain the information data.

Remarks
The information data is specified in the glove user's manual. The size of the information block is always 32 bytes.

`void fdGetDriverInfo(fdGlove *pFG, unsigned char *pData)`

Obtains the information data block of the driver.

Return value
None.

Parameters
`pFG`
Pointer to a glove device. This is the value returned by `fdOpen()`.

`pData`
Array of 32 bytes that will contain the information data.

Remarks
The information data is a zero terminated string that contains driver information. The size of the information block is always 32 bytes.

5. Gesture definitions

The currently defined set of gestures is comprised of binary open/close configurations of the fingers excluding the thumb. There are $2^4 = 16$ such possible combinations. Gesture number 0 is defined as all the fingers (excluding the thumb) being closed, and gesture number 15 as all the fingers open. The index finger indicates the least significant bit. For example, the index finger point gesture will therefore be number 1, and the little finger point gesture number will be 8. An invalid (unrecognizable) gesture is defined as the value -1.

A scaled sensor value of higher than the upper threshold setting will indicate a closed finger, while a scaled sensor value of lower than the lower threshold setting will indicate an open finger. A value in-between is invalid and will result in an invalid gesture. In the case of multiple finger joint angle measurements (such as the 16 sensor glove), the maximum of the individual joint sensor values is taken to obtain a closed gesture and the minimum to obtain an open gesture. Closed gestures take precedence, in other words, bending only one joint of a finger will count as a closed gesture.

The built-in gesture recognition capabilities of the glove driver is limited in scope and user independence. High level gesture recognition algorithms that rely on training sequences are suggested for advanced applications.

Finger:	Little	Ring	Middle	Index
5DT Data Glove 5 sensor:	E	D	C	B
5DT Data Glove 16 sensor:	12,13	9,10	6,7	3,4
Driver sensor index:	12,13#	9,10#	6,7#	3,4#

Gesture Number	Flexure (0=flexed, 1=unflexed)				Gesture Description	Fig.
0	0	0	0	0	Fist	3.0
1	0	0	0	1	Index finger point	3.1
2	0	0	1	0	Up yours	3.2
3	0	0	1	1	Two finger point	3.3
4	0	1	0	0	Ring finger point	3.4
5	0	1	0	1	Ring index point	3.5
6	0	1	1	0	Ring middle point	3.6
7	0	1	1	1	Three finger point (or not little point)	3.7
8	1	0	0	0	Little finger point	3.8
9	1	0	0	1	Howzit	3.9
10	1	0	1	0	Little middle point	3.10
11	1	0	1	1	Not ring finger point	3.11
12	1	1	0	0	Little ring point	3.12
13	1	1	0	1	Not up yours	3.13
14	1	1	1	0	Not index finger point	3.14
15	1	1	1	1	Flat hand	3.15

Table 3 - Gesture definition scheme as implemented for the 5DT Data Glove Driver

When the 5DT Data Glove 5 is used, both these driver sensor indices will return the same value.
When the 5DT Data Glove 16 is used, the driver sensor indices will return different values. The maximum of the two values will be used to test for a flexed (closed) gesture, and the minimum value of the two will be used to test for an unflexed (open) gesture.

The following gestures are currently defined (right hand shown):

Figure 3 - Gesture Illustrations

6. Auto-calibration

The driver can provide sensor outputs in an automatic, linearly calibrated fashion. During every update, the raw value read from the sensor is compared to the current minimum and maximum raw values (raw_{min} and raw_{max}) as set by the functions `fdSetCalibrationAll()`, `fdSetCalibration()` or `fdResetCalibration()`. If the current minimum and maximum values are exceeded, they are overwritten. The upper and lower calibration values are therefore continuously pushed "outwards". The normalized output is given by the first order equation

$$out = \frac{raw_{val} - raw_{\min}}{raw_{\max} - raw_{\min}} . Max,$$

which is in [0...Max]. The value of Max is set by the functions `fdSetSensorMaxAll()` and `fdSetSensorMax()`. Doing a few flexing movements with the hand quickly sets the operating values for raw_{min} and raw_{max}, and calibrates the glove.

The auto-calibration process can be ignored by simply regarding only the raw sensor outputs. It would be up to the application developer to provide a suitable calibration process. Note that calibration is *mandatory*, especially with the 16 sensor glove which contains no hardware calibration possibilities (miniature preset potentiometers).