DTAPI 5.0

RELEASE NOTES

SDK

April 2012

Table of Contents

1. DTAPI 5.0 – Overview

DTAPI 5.0 is a major new release of the API for DekTec PCI, PCI Express, USB and IP devices. The high-level reasons for this new release include the following:

- One common code base for all parts of the Windows and Linux SDK: drivers, user-space library (DTAPI) and service. This way, new hardware from DekTec can be supported on Windows and Linux right from the start. Most bug fixes will be automatically available on both platforms.
- XML-based description of hardware devices in the driver. This simplifies the addition of new hardware devices, as most device-specific features can be described in the XML.
- Better plug-and-play and power-management support.
- Smart architecture that is ready for new DTAPI features.

From a programmer's perspective the following has changed:

- New mechanism to configure I/O ports;
- Extension of the capability mechanism; capabilities are linked to I/O configuration options;
- New functions for handling driver events;
- API clean up for a more consistent DTAPI;
- Removal of rarely-used/obsolete functions.

2. Changes per Category

2.1. Helper Structures

Structure	Action	Remarks
`DtHwFuncDesc`	*`m_StreamType`* removed	Stream type is encoded in the port capabilities in *`m_Flags`*

2.2. Global DTAPI Functions

Function	Action	Remarks
`DtapiGetDevice DriverVersion`	Removed 1 overload	Device-driver category must always be speci-fied
`DtapiGetVersion`	Removed 1 overload	Four version numbers (major, minor, bugfix, build) must always be retrieved
`DtapiPciScan`	Removed	Use **`DtapiHwFuncScan`**

2.3. DtDevice

Function	Action	Remarks
`GetDeviceDriverVersion`	Remove overload with 2 arguments	Use overload with 4 arguments
`GetIoConfig`	New function arguments	See DtDevice::SetIoConfig
`I2CLock, I2CUnlock, I2CRead, I2CWrite`	Moved to DtInpChannel	
`RegisterCallback`	Added	Register callbacks for handling events
`SetIoConfig`	New function arguments	Entirely new I/O configuration interface
`UnregisterCallback`	Added	

2.4. DtDemodControl

Removed. Functions from **`DtDemodControl`** are moved to **`DtInpChannel`**.

`DtInpChannel::AttachToPort` is extended with an extra argument *`Exclusive`*, which can be set to *false* to emulate the behaviour of the old **`DtDemodControl`**. This way, multiple **`DtInpChannel`** objects can have shared access to demodulators.

Function	Action	Remarks
`GetDemodStatus`	Removed	Use **`DtInpChannel::GetStatistics`**

2.5. DtInpChannel

Function	Action	Remarks
`Attach`	Removed	Use **`AttachToPort`**
`AttachToPort`	Argument *Exclusive* added	
`DetectIoStd`	Added	
`ClearFifo`	Changed	Removed *`SubCh`* parameter
`GetFifoLoad`	Changed	Removed *`SubCh`* parameter
`GetIoConfig`	Added	

GetMaxFifoSize	Changed	Removed *SubCh* parameter
GetReceiveByteCount	Removed	
GetRfLevel	Removed	Use **DtInpChannel::GetStatistics** with statistic **DTAPI_STAT_RFLVL_CHAN**
GetRxMode	Changed	See **SetRxMode**
GetStatistics(int&)	Removed	Use **GetViolCount(int& ViolCount)**
ReadDirect	Removed	Use **Read**
ReadSubCh	Removed	ADC data can be read
ReadUsingDma	Removed	Use **Read**
SetIoConfig	Added	
SetLoopBackMode	Removed	Not useful for user applications
SetRxControl	Changed	Removed *SubCh* parameter
SetRxMode	Changed	ASI/SDI mode must be set with **SetIoConfig** **DTAPI_RX_TIMESTAMP32/64** renamed to **DTAPI_RXMODE_TIMESTAMP32/64** **DTAPI_RX_TIMESTAMP** is obsolete
WriteLoopBackData	Removed	Not useful for user applications

2.6. DtOutpChannel

Function	Action	Remarks
Attach	Removed	Use **AttachToPort**
GetChannelType	Removed	Superfluous function
GetIndexOnDevice	Removed	Deprecated
GetIoConfig	Added	
GetModControl	Add parameter *pXtraPars*	
GetRfControl	New symbolic values for *LockStatus*	Status of individual PLLs can be requested
GetSpiClk	Added	For DTA-2142 to get SPI fixed clock
GetTransmitByteCount	Removed	Deprecated
GetTsRateBps	Removed overload with *ClockGenMode*	External clock mode is now handled by I/O configuration **TSRATESEL**
ReadLoopBackData	Removed	Not useful for user applications
SetCustomRollOff	Added	For DTA-2107
SetIoConfig	Added	
SetLoopBackMode	Removed	Not useful for user applications
SetSpiClk	Added	For DTA-2142 to set SPI fixed clock
SetTsRateBps	Removed overload with *ClockGenMode*	External clock mode is now handled by I/O configuration **TSRATESEL**
WriteDirect	Removed	Use Write
WriteUsingDma	Removed	Use Write

3. Capabilities

3.1. Definition

The capability system has been extended significantly.

Capability	Identifies a characteristic or feature of a physical port. DTAPI type: **`DtCaps`** (can be OR-ed together)
I/O Capability	Capability that is linked to I/O configuration: If an I/O capability is supported, **`SetIoConfig`** can be used to enable the port feature.
Standard Capability	These capabilities indicate whether a certain function is supported by the port, and are unrelated to I/O configuration.

3.2. Organization

Capabilities are organized in *groups*, *capabilities* and *subcapabilities*. The table below lists all capability groups.

Group	I/O Capability?	Description
`BOOLIO`	Yes	Boolean I/O capabilities, e.g. **`FAILSAFE`** which indicates whether a port supports a failsafe mode
`DEMODPROPS`	No	Demodulator properties
`FREQBAND`	No	Frequency band, e.g. **`LBAND`**
`IODIR`	Yes	The direction of the signal flow: **`INPUT`**, **`OUTPUT`** or **`DISABLED`**. The sub capabilities in this group indicate how a physical port is connected to the input or output channel. This encodes features like double buffering.
`IOPROPS`	No	Miscellaneous capabilities that don't fit elsewhere
`IOSTD`	Yes	The I/O standard used on this port. The most important ones are: **`ASI`**, **`SDI`**, **`SPI`**, **`DEMOD`**, **`IP`** and **`MOD`**
`MODSTD`	No	Supported modulation standards; Used both for modulators and demodulators/receivers
`MODPROPS`	No	Other capabilities (besides **`MODSTD`**) related to modulation
`RFCLKSEL`	Yes	Modulator RF clock - Selection of reference source: internal or external
`TSRATESEL`	Yes	Capabilities in this group selects between ways to generate the transport-stream rate
Groups that apply to DTA-2142 only:		
`SPICLKSEL`	Yes	Parallel port clock - Selection of reference source: internal or external
`SPIMODE`	Yes	Mode to operate a 25-pin sub-D parallel port
`SPISTD`	Yes	I/O standard used on 25-pin sub-D parallel port

For a complete list of capabilities, please refer to the **`DTAPI_CAP_XXX`** constants in DTAPI.h

3.3. Usage of Capabilities

The global DTAPI function **`::DtapiHwFuncScan`** scans the hardware and creates a hardware function descriptor (**`DtHwFuncDesc`**) for each port. Capabilities are encoded in member *`m_Flags`*.

To test for a certain capability:

```
if ((HwFuncDesc.m_Flags & DTAPI_CAP_ASI) != 0)
   // Port supports ASI
```

4. I/O Configuration

4.1. SetIoConfig and GetIoConfig

Use the `SetIoConfig` to set the I/O configuration of a port, and `GetIoConfig` to read it back.

On Windows, the I/O configuration settings are persisted in the registry. After a reboot, the I/O configurations will be automatically restored to the last settings. On Linux, the application is responsible for configuring the ports.

4.2. Relation to Capabilities

`SetIoConfig` and `GetIoConfig` have four parameters to identify a particular I/O configuration option:

Set/GetIoConfig parameter	Linked to
Port	Physical port number
Group	Capability group (Only I/O capabilities have a corresponding I/O configuration group)
Value	Capability
SubValue	Subcapability

Example:

An output port that can be configured in double buffered mode has **`DTAPI_CAP_DBLBUF`**, a subcapability of capability **`DTAPI_CAP_OUTPUT`** in the **`IODIR`** group. To configure a port as double buffered, use:

```
    Dvc.SetIoConfig(Port, DTAPI_IOCONFIG_IODIR,       // Group
                          DTAPI_IOCONFIG_OUTPUT,      // Value
                          DTAPI_IOCONFIG_DBLBUF);     // Subvalue
```

For a complete list of I/O configuration groups, values and subvalues, see the **`DTAPI_IOCONIG_XXX`** constants in DTAPI.h

4.3. SetIoConfig Variants

Two `SetIoConfig` functions are defined, one at device level and one at channel level. The I/O configuration of a port at device level can only be changed when the port is not used (no channel object attached). Some, but not all, I/O configuration changes can be performed at channel level while the channel object is attached to the hardware.

The driver validates whether the I/O configurations of multiple port are consistent with each other. For example, on the DTA-2137 only one port can be set to **`DTAPI_IOCONFIG_SWS2APSK`**, otherwise an error is returned.

To simplify configuration changes that must be done in a specific order and to prevent temporary invalid configurations a new `SetIoConfig` "transaction" variant is introduced. With the transaction variant the IO configuration only needs to be valid before and after the complete transaction, not after each individual configuration item.

5. Changes in DTAPI Functionality

5.1. SetRxMode and SetTxMode for ASI / SDI Ports

In DTAPI versions prior to v5.0, `SetRxMode` and `SetTxMode` could be used to select between operation in ASI mode and operation in SDI mode.

Starting from DTAPI v5.0, selection between ASI and SDI modes is moved from `SetRxMode` / `SetTxMode` to the I/O configuration system. An application can check whether a port supports ASI and/or SDI with the capabilities `DTAPI_CAP_ASI` and `DTAPI_CAP_SDI` respectively, which are in the I/O standard group (`IOSTD`).

To configure a port in ASI mode:

```
  if ((HwFuncDesc.m_Flags & DTAPI_CAP_ASI) != 0)
     Dvc.SetIoConfig(Port, DTAPI_IOCONFIG_IOSTD, DTAPI_IOCONFIG_ASI);
```

5.2. GetStatistics Functionality for Demodulators

DTAPI v5.0 uses a new class to represent receiver measurements and statistics: `DtStatistic`. A summary of its declaration is shown below. Refer to DTAPI.h for the full definition.

```
struct DtStatistic
{
    DtStatistic();
    DtStatistic(int StatisticId);    // Constructor with DTAPI_STAT_xxx initialization

    enum StatValueType { STAT_VT_UNDEFINED, STAT_VT_BOOL, STAT_VT_DOUBLE, STAT_VT_INT };
    DTAPI_RESULT  m_Result;          // Result of retrieving the statistic
    int  m_StatisticId;              // Identifies the statistic: DTAPI_STAT_XXX
    StatValueType  m_ValueType;      // Value type of statistic: STAT_VT_XXX
    union {
        bool  m_ValueBool;           // Value if value type is STAT_VT_BOOL
        double  m_ValueDouble;       // Value if value type is STAT_VT_DOUBLE
        int  m_ValueInt;             // Value if value type is STAT_VT_INT
    };
    DTAPI_RESULT  GetName(..), GetValue(..), SetId(..);
};
```

Statistics are identified by its ID (`m_StatisticId`). See DTAPI.h for a list of `DTAPI_STAT_X` identifiers. The function `GetName()` returns a full name and a short name of the statistic. The value of the statistic can be retrieved with `GetValue()`.

Note: If the type of a statistic is `STAT_VT_INT`, its value can be retrieved both as `int` and as `double`.

The following statistics functions are available:

Function	Description

`GetStatistic(int, int&)`
`GetStatistic(int, double&)`
`GetStatistic(int, bool&)`
	Return a single statistic

`GetStatistics(int, DtStatistic*)`	Return an array of statistics
`GetSupportedStatistics(int&, DtStatistic*)`	Returns all supported statistics on a port

5.3. ADC Sub Channels

The sub-channel parameter as found in some `DtInpChannel` functions has been removed. The associated identifiers `DTAPI_SUBCH_MAIN` and `DTAPI_SUBCH_ADC` have been removed, too.

In the new interface, port 3 of DTA-2135 is used as a 'virtual' port to get a stream of IF samples. This port has capabilities **`DTAPI_CAP_INPUT`** and **`DTAPI_CAP_IFADC`**.