Document	: FLEXCOP_SYS_V100
Author	: M. Majoor
Subject	: FLEXCOP generic driver

Revision overview

Revision/Date	Author	Description
200504010	MM	Initial draft

Communication with the FLEXCOP.SYS driver takes place using DeviceIO calls. This document describes these calls. Only the function and parameters passed/returned are discussed.

The driver will de-allocate all allocated resources only when the driver is shut down. This means that allocated memory keeps being allocated if not properly de-allocated. This also means that multiple instances, which use the driver, can use the same allocated memory.

The driver is interrupt oriented. This means that automated actions are triggered only by means of an FLEXCOP interrupt. The interrupt handling mechanism, sets the way how an interrupt is handled. The following actions are possible for every single FLEXCOP interrupt:
- an ICTL_WAIT_NOTIFY is 'acknowledged' (eventActive)
- specific data is written to a specified FLEXCOP register (signalActive)
- specific data is copied to a FIFO buffer

The driver can use buffers to store data when an interrupt is activated. Although the term FIFO (First In First Out) is used here, it is actually not true. This is because retrieving the data from the buffers is not under control of the driver. The application reading the buffers is responsible for reading the correct buffer.
Although primarily used to buffer data from a single memory source, it can also be used to acquire data from multiple sources.

IOCTL_GET_VERSION Get version of driver.		
IN	-	
OUT	USHORT majorversion	Major version
	USHORT minorversion	Minor version
	ULONG build	Build date $YYYYMMDD YYYY = Year MM = Month DD = Day

IOCTL_GET_STATUS Get status information.		
IN	ULONG DMAbuffer	DMA buffer to return status for.
OUT	ULONG interrupts	Interrupt occurs (total interrupts for driver).
	ULONG isr	Interrupt status register of last occurred interrupt. (FLEXCOP register $20C)
	PVOID vaBuffer	Virtual address DMA buffer. You need this address when FIFO buffers are to be used. FIFO buffers can only reference to memory allocated for DMA.
	PHYSICAL_ADDRESS paBuffer	Physical address DMA buffer.
	ULONG buffersize	Size of DMA buffer in bytes.
	ULONG fifoOverflows	Global number of FIFO overflows. An overflow is indicated when a FIFO buffer is written to by the driver, without having the FIFO buffer read by an application.

IOCTL_FLEXCOP_READ Read register from FLEXCOP (dword access).		
IN	ULONG address	Address of the register from the FLEXCOP to read.
OUT	ULONG data	Data read from register.

IOCTL_FLEXCOP_WRITE Write data to a register of the FLEXCOP (dword access).

IN	ULONG address	Address of the register from the FLEXCOP to write to.
	ULONG data	Data to write to the register.
OUT	-	

IOCTL_FLEXCOP_READ2		
Read register from FLEXCOP (word access).		
IN	ULONG address	Address of the register from the FLEXCOP to read.
OUT	USHORT data	Data read from register.

IOCTL_FLEXCOP_WRITE2		
Write data to a register of the FLEXCOP (word access).		
IN	ULONG address	Address of the register from the FLEXCOP to write to.
	USHORT data	Data to write to the register.
OUT	-	

IOCTL_WAIT_NOTIFY		
Wait for a notification (interrupt). Only when an interrupt, which must have been 'activated', is generated this call will return. Thus, this call waits until this occurs. The only way to recover for this, besides an actual interrupt, is to manually trigger an event using IOCTL_GENERATE_EVENT. Note: the driver only supports a single IOCTL_WAIT_NOTIFY. If more than a single IOCTL_WAIT_NOTIFY is issued only the first is 'acknowledged'. The others will wait for the next generated event,		
IN	-	
OUT	ULONG bogus	Nothing is actually being returned.

IOCTL_GENERATE_EVENT		
Generate a manual event. Used to end an IOCTL_WAIT_NOTIFY manually.		
IN	ULONG bogus	Not used.
OUT	-	

IOCTL_DMA_ALLOCATE
Allocate memory which can be used for DMA purposes. The memory allocated is a single contiguous block of memory. The driver allows a maximum of 256 buffers to be allocated.

IN	ULONG buffersize	Size in bytes of contiguous memory to allocate.
OUT	LONG bufferId	Identification of buffer. This identification is used to de-allocate the memory or getting status information for it.
	PVOID vaBuffer	Virtual address of buffer. This address is to be used when FIFO buffers are allocated. These FIFO buffers are only allowed to use the memory allocated using this call. Note: Windows 98 allows using this address to access the allocated memory directly. However, this does not work with W2000 (use IOCTL_DMA_READ). It is advised, for Windows 98 also, to use IOCTL_DMA_READ and IOCTL_DMA_WRITE to access the allocated memory.
	PHYSICAL_ADDRESS paBuffer	Physical address of buffer.
	ULONG buffersize	Size in bytes of buffer

IOCTL_DMA_RELEASE		
Release the memory previously allocated. FIFO buffers using the memory being released are also being de-allocated.		
IN	LONG bufferId	Identification of buffer. This is the identification returned when IOCTL_DMA_ALLOCATE is used.
OUT	-	

IOCTL_DMA_READ		
Read from IOCTL_DMA_ALLCOATE allocated memory.		
IN	LONG bufferId	Identification of buffer. This is the identification returned when IOCTL_DMA_ALLOCATE is used.
	PUCHAR bufferTransfer	Pointer to target buffer
	ULONG bufferSourceIndex	Index in bytes into source (DMA memory) to start copying from.
	ULONG bufferTargetIndex	Index in bytes into target buffer to start copying to.
	ULONG bufferTransferLength	Number of bytes to transfer.
OUT	-	

IOCTL_DMA_WRITE
Write to IOCTL_DMA_ALLCOATE allocated memory.

IN	LONG bufferId	Identification of buffer. This is the identification returned when IOCTL_DMA_ALLOCATE is used.
	PUCHAR bufferTransfer	Pointer to source buffer
	ULONG bufferSourceIndex	Index in bytes into source buffer to start copying from.
	ULONG bufferTargetIndex	Index in bytes into target buffer (DMA memory) to start copying to.
	ULONG bufferTransferLength	Number of bytes to transfer.
OUT	-	

IOCTL_FIFO_ALLOCATE		
Allocate a number of FIFO buffers of the indicated size. A FIFO uses (part of) allocated memory (IOCTL_DMA_ALLOCATE) as it's source data. The driver can handle a maximum of 256 FIFO buffers.		
IN	LONG bufferId	-
	LONG buffers	Number of FIFO buffers to allocate. These all use the same memory as source. When a FIFO buffer is written to then the next FIFO in the list will be used (assuming buffers > 1). When the last FIFO buffer is written to then the first FIFO buffer is used as next target.
	Array[0..1] of PVOID bufferTransfer	Address of source data (data which is to be copied to the FIFO buffers). This address must be within the range of driver allocated memory (i.e. allocated with IOCTL_DMA_ALLOCATE). For each sub buffer a different source address can be set. The selection of which index to use is done by the driver (based on which buffer is being written when an interrupt occurs). Note: It is essential that the application writes or reads the sub buffer addresses at least once (eg. addresses $000 and $0010) since these values are preserved and used in the process. If the sub buffer addresses are never used then the sub buffer addresses are unknown (the driver does not read it specifically).
	ULONG bufferTransferLength	Size in bytes of the memory to be copied. This is also the size of the FIFO buffers being allocated.
	BOOLEAN bufferWritten	-
	ULONG bufferOrder	-
	ULONG bufferOverflows	-
	ULONG bufferIrqs	-
	ULONG bufferIrqOverflows	-
OUT	LONG bufferId	Identifier of first FIFO buffer allocated.

	LONG buffers	-
	PVOID array[0..1] of bufferTransfer	-
	ULONG bufferTransferLength	-
	BOOLEAN bufferWritten	-
	ULONG bufferOrder	-
	ULONG bufferOverflows	-
	ULONG bufferIrqs	-
	ULONG bufferIrqOverflows	-

IOCTL_FIFO_RELEASE De-allocate a number of FIFO buffers. Note that all FIFO's using the same interrupt as source will be deactivated.		
IN	LONG bufferId	FIFO buffer identification (first buffer) to release. The driver will make sure that an invalid entry will be within the allowed range. This means that a negative value will effectively use the very first buffer. A very high value will use the last buffer. To release all FIFO buffers use -1.
	LONG buffers	Number of FIFO buffers to release. The driver will make sure that only the correct amount of buffers to release are used. To release all FIFO buffers use 256 (or a higher value). Note: you should typically use the same number of buffers as have been allocated.
	PVOID array[0..1] of bufferTransfer	-
	ULONG bufferTransferLength	-
	BOOLEAN bufferWritten	-
	ULONG bufferOrder	-
	ULONG bufferOverflows	-
	ULONG bufferIrqs	-
	ULONG bufferIrqOverflows	-
OUT	LONG bufferId	FIFO buffer identification of first buffer released.
	LONG buffers	Number of FIFO buffers actually released.

	PVOID array[0..1] of bufferTransfer	-
	ULONG bufferTransferLength	-
	BOOLEAN bufferWritten	-
	ULONG bufferOrder	-
	ULONG bufferOverflows	-
	ULONG bufferIrqs	-
	ULONG bufferIrqOverflows	-

IOCTL_FIFO_READ Read data from a FIFO buffer. When a FIFO buffer is read it will be marked 'read'. If the driver writes data into the FIFO buffer it is marked as being 'written'. When a buffer was already marked 'written' then an overflow is the result.		
IN	LONG bufferId	FIFO buffer identification to read data from.
	LONG buffers	-
	PVOID array[0..1] of bufferTransfer	bufferTransefr[0] should point to the target buffer to where data from the FIFO buffer is copied to.
	ULONG bufferTransferLength	Size of target buffer. Must be at least the size of the allocated FIFO buffer. The whole FIFO buffer is always being copied.
	BOOLEAN bufferWritten	-
	ULONG bufferOrder	-
	ULONG bufferOverflows	-
	ULONG bufferIrqs	-
	ULONG bufferIrqOverflows	-
OUT	LONG bufferId	-
	LONG buffers	-
	PVOID array[0..1] of bufferTransfer	-
	ULONG bufferTransferLength	Number of actual bytes transferred. This is always the size of the allocated FIFO buffer.
	BOOLEAN bufferWritten	Indicates if the FIFO contains valid data. If FALSE it indicates that the FIFO has not been written to since it has been read.

	ULONG bufferOrder		Order number of the buffer being written. Every time a FIFO buffer is written to it gets a number that is increased for every write using the same interrupt source.
	ULONG bufferOverflows		Overflows counted on this particular FIFO buffer.
	ULONG bufferIrqs		Total number of interrupts generated for the interrupt source as used by the FIFO.
	ULONG bufferIrqOverflows		Overflows counted on FIFO's using the same interrupt source.

IOCTL_IRQ_WRITE

Write interrupt handling mechanism.
This defines the interrupt handling. An interrupt can be set to generate an event; set a FLEXCOP register to a defined state; copy data to a FIFO buffer.

IN	LONG IrqId		FLEXCOP interrupt number (0..11). See the FLEXCOP interrupt status register for the types.
	IRQBUFFER IrqInformation	ULONG irqsReceived	Number of interrupts detected of this type (whether active or not).
		ULONG irqsActiveReceived	Number of interrupts detected of this type when interrupt is active.
		BOOLEAN active	Activates the interrupt or not. This is the global activation of the interrupt. When FALSE the other activations are not used.
		BOOLEAN eventActive	Indicates if a call which waits for an event is to be notified (IOCTL_WAIT_NOTIFY).
		BOOLEAN signalActive	Indicates that the signaling mechanism is to be used. The signaling mechanism will write specific data to a FLEXCOP register.
		BOOLEAN fifoActive	Indicates that FIFO buffering is to be used.
		BOOLEAN Reserved	
		ULONG signalRegister	Register to read and write back when the signaling mechanism is active (signalActive = TRUE)
		ULONG signalAnd	The AND value applied to the contents as read from the register (signalRegister).
		ULONG signalOr	The OR value applied to the ANDed contents as read from the register (signalRegister).
		ULONG signalXor	The XOR (invert) value applied to the ANDed and ORed contents as read from the register (signalRegister).

		UCHAR fifoBufferPrevious	The last written FIFO buffer.
		UCHAR fifoBufferFirst	The first FIFO buffer to write data to when an interrupt is generated.
		UCHAR fifoBufferLast	The last FIFO buffer to write data to when an interrupt is generated. Must be >= fifoBufferFirst.
		ULONG fifoBufferCirculated	Counter incremented when the first FIFO buffer is used.
		ULONG fifoOverflows	Counter incremented when an overflow is detected. An overflow is detected when a FIFO buffer is written to without that it has been read (i.e. bufferWritten flag is used for this)
OUT	-		

IOCTL_IRQ_READ Read interrupt handling mechanism.			
IN	LONG IrqId		FLEXCOP interrupt number (0..11). See the FLEXCOP interrupt status register for the types.
	IRQBUFFER IrqInformation	-	-
OUT	LONG IrqId		FLEXCOP interrupt number (0..11). See the FLEXCOP interrupt status register for the types.
	IRQBUFFER IrqInformation	ULONG irqsReceived	Number of interrupts detected of this type (whether active or not).
		ULONG irqsActiveReceived	Number of interrupts detected of this type when interrupt is active.
		BOOLEAN active	Activates the interrupt or not. This is the global activation of the interrupt. When FALSE the other activations are not used.
		BOOLEAN eventActive	Indicates of a call which waits for an event is to be notified (IOCTL_WAIT_NOTIFY).
		BOOLEAN signalActive	Indicates that the signaling mechanism is to be used. The signaling mechanism will write specific data to a FLEXCOP register.
		BOOLEAN fifoActive	Indicates that FIFO buffering is to be used.

		BOOLEAN Reserved	
		ULONG signalRegister	Register to read and write back when the signaling mechanism is active (signalActive = TRUE)
		ULONG signalAnd	The AND value applied to the contents as read from the register (signalRegister).
		ULONG signalOr	The OR value applied to the ANDed contents as read from the register (signalRegister).
		ULONG signalXor	The XOR (invert) value applied to the ANDed and ORed contents as read from the register (signalRegister).
		UCHAR fifoBufferPrevious	The last written FIFO buffer.
		UCHAR fifoBufferFirst	The first FIFO buffer to write data to when an interrupt is generated.
		UCHAR fifoBufferLast	The last FIFO buffer to write data to when an interrupt is generated. Must be >= fifoBufferFirst.
		ULONG fifoBufferCirculated	Counter incremented when the first FIFO buffer is used.
		ULONG fifoOverflows	Counter incremented when an overflow is detected. An overflow is detected when a FIFO buffer is written to without that it has been read (i.e. bufferWritten flag is used for this)