This page describes the structures and procedures used by the cx2341x DMA
engine.

Introduction
============

The cx2341x PCI interface is busmaster capable. This means it has a DMA
engine to efficiently transfer large volumes of data between the card and main
memory without requiring help from a CPU. Like most hardware, it must operate
on contiguous physical memory. This is difficult to come by in large quantities
on virtual memory machines.

Therefore, it also supports a technique called "scatter-gather". The card can
transfer multiple buffers in one operation. Instead of allocating one large
contiguous buffer, the driver can allocate several smaller buffers.

In practice, I've seen the average transfer to be roughly 80K, but transfers
above 128K were not uncommon, particularly at startup. The 128K figure is
important, because that is the largest block that the kernel can normally
allocate. Even still, 128K blocks are hard to come by, so the driver writer is
urged to choose a smaller block size and learn the scatter-gather technique.

Mailbox #10 is reserved for DMA transfer information.

Note: the hardware expects little-endian data ('intel format').

Flow
====

This section describes, in general, the order of events when handling DMA
transfers. Detailed information follows this section.

- The card raises the Encoder interrupt.
- The driver reads the transfer type, offset and size from Mailbox #10.
- The driver constructs the scatter-gather array from enough free dma buffers
  to cover the size.
- The driver schedules the DMA transfer via the ScheduleDMAtoHost API call.
- The card raises the DMA Complete interrupt.
- The driver checks the DMA status register for any errors.
- The driver post-processes the newly transferred buffers.

NOTE! It is possible that the Encoder and DMA Complete interrupts get raised
simultaneously. (End of the last, start of the next, etc.)

Mailbox #10
===========

The Flags, Command, Return Value and Timeout fields are ignored.

Name: Mailbox #10
Results[0]: Type: 0: MPEG.
Results[1]: Offset: The position relative to the card's memory space.
Results[2]: Size: The exact number of bytes to transfer.

My speculation is that since the StartCapture API has a capture type of "RAW"
available, that the type field will have other values that correspond to YUV

and PCM data.

Scatter-Gather Array
====================

The scatter-gather array is a contiguously allocated block of memory that tells the card the source and destination of each data-block to transfer. Card "addresses" are derived from the offset supplied by Mailbox #10. Host addresses are the physical memory location of the target DMA buffer.

Each S-G array element is a struct of three 32-bit words. The first word is the source address, the second is the destination address. Both take up the entire 32 bits. The lowest 18 bits of the third word is the transfer byte count. The high-bit of the third word is the "last" flag. The last-flag tells the card to raise the DMA_DONE interrupt. From hard personal experience, if you forget to set this bit, the card will still "work" but the stream will most likely get corrupted.

The transfer count must be a multiple of 256. Therefore, the driver will need to track how much data in the target buffer is valid and deal with it accordingly.

Array Element:

- 32-bit Source Address
- 32-bit Destination Address
- 14-bit reserved (high bit is the last flag)
- 18-bit byte count

DMA Transfer Status
===================

Register 0x0004 holds the DMA Transfer Status:

Bit
0   read completed
1   write completed
2   DMA read error
3   DMA write error
4   Scatter-Gather array error