# Emulator - Emulator Issues #10265

## Star Wars: The Clone Wars hangs on loading screen with DSP-HLE and JIT Recompiler

05/06/2017 11:09 PM - ligfx

| | | | |
|---|---|---|---|
| **Status:** | Fixed | **% Done:** | 0% |
| **Priority:** | Normal | | |
| **Assignee:** | | | |
| **Category:** | | | |
| **Target version:** | | | |
| **Operating system:** | N/A | **Relates to performance:** | No |
| **Issue type:** | Bug | **Easy:** | No |
| **Milestone:** | | **Relates to maintainability:** | No |
| **Regression:** | No | **Regression start:** | |
| **Relates to usability:** | No | **Fixed in:** | 5.0-3860 |

**Description**

**Game Name?**

Star Wars - The Clone Wars - GSXE64

**What's the problem? Describe what went wrong.**

The game hangs at the very first loading screen.

**What steps will reproduce the problem?**

Override Dolphin's built-in game settings to force DSP-HLE, then run using the JIT Recompiler.

**Which versions of Dolphin did you test on? Does using an older version of Dolphin solve your issue? If yes, which versions of Dolphin used to work?**

Tested on 5.0-3635. Dolphin 5.0 exhibits issue https://bugs.dolphin-emu.org/issues/1132 instead. JMC47 said it used to boot before phire's coretiming fixes.

**What are your PC specifications?** (CPU, GPU, Operating System, more)

Intel Core i7 @ 2.8 GHz, Intel Iris 1536 MB, macOS 10.12.4

**Is there any other relevant information? (e.g. logs, screenshots, configuration files)**

Could be related to https://bugs.dolphin-emu.org/issues/10240

### History

**#1 - 05/06/2017 11:17 PM - ligfx**

Various exploratory notes so far:

The game uses the AX ucode. It repeatedly executes the following command list:

```
0000 // CMD_SETUP
8063
cac0
0007 // CMD_SET_LR
8063
87a0
0002 // CMD_PB_ADDR
8063
cb00
```

```
0003 // CMD_PROCESS
0012 // CMD_UNK_12
8000
000a
8037
07c0
000e // CMD_OUTPUT
8063
87a0
8063
8520
```

Of possible note is the unimplemented CMD_UNK_12.

I also noted that, after switching to the AX ucode, all values read from and written to the DSP control register are too large by 0x0800 / should be AND-ed against ~DSP_CONTROL_MASK (0x0C07). However, hacking around this to make the values identical with DSP-LLE didn't seem to change anything.

#### #2 - 05/07/2017 12:38 AM - ligfx

The issue can be resolved by removing the lines

```
// If this event needs to be scheduled before the next advance(), force one early
if (!s_is_global_timer_sane)
  ForceExceptionCheck(cycles_into_future);
```

from CoreTiming.cpp. I haven't had the time yet to look into why this only triggers in the JIT recompiler / DSP-HLE combination, nor what else it breaks.

#### #3 - 05/08/2017 12:53 AM - ligfx

More exploration:

Disabling ForceExceptionCheck for only events of type DSPint fixes the issue. When starting the game, here's the list of events going through that codepath on HLE + JIT Recompiler:

```
ForceExceptionCheck #1 ARAMint 0x00 - cycles=246, downcount 5499 -> 246, slice length 7691 -> 2438
ForceExceptionCheck #2 ARAMint 0x00 - cycles=246, downcount 2089 -> 246, slice length 7717 -> 5874
ForceExceptionCheck #3 SyncGPUCallback 0x3e8 - cycles=1000, downcount 1044 -> 1000, slice length 7716 -> 7672
```

```
ForceExceptionCheck #4 SyncGPUCallback 0x3e8 - cycles=1000, downcount 7074 -> 1000, slice length 7102 -> 1028
ForceExceptionCheck #5 SyncGPUCallback 0x3e8 - cycles=1000, downcount 5003 -> 1000, slice length 7721 -> 3718
ForceExceptionCheck #6 SyncGPUCallback 0x3e8 - cycles=1000, downcount 4425 -> 1000, slice length 15442 -> 1201
7
ForceExceptionCheck #7 ARAMint 0x00 - cycles=246, downcount 1442 -> 246, slice length 5704 -> 4508
ForceExceptionCheck #8 ARAMint 0x00 - cycles=246, downcount 1195 -> 246, slice length 1195 -> 246
ForceExceptionCheck #9 ARAMint 0x00 - cycles=246, downcount 948 -> 246, slice length 948 -> 246
ForceExceptionCheck #10 ARAMint 0x00 - cycles=246, downcount 702 -> 246, slice length 702 -> 246
ForceExceptionCheck #11 ARAMint 0x00 - cycles=246, downcount 456 -> 246, slice length 456 -> 246
ForceExceptionCheck #12 ARAMint 0x00 - cycles=246, downcount 15290 -> 246, slice length 15317 -> 273
ForceExceptionCheck #13 AICallback 0x00 - cycles=0, downcount 14425 -> 0, slice length 15042 -> 617
ForceExceptionCheck #14 DSPint 0x80 - cycles=0, downcount 8287 -> 0, slice length 15441 -> 7154
ForceExceptionCheck #15 DSPint 0x08 - cycles=200, downcount 7173 -> 200, slice length 8282 -> 1309
ForceExceptionCheck #16 DSPint 0x80 - cycles=0, downcount 6426 -> 0, slice length 15416 -> 8990
ForceExceptionCheck #17 DSPint 0x80 - cycles=0, downcount 8867 -> 0, slice length 15440 -> 6573
ForceExceptionCheck #18 DSPint 0x80 - cycles=0, downcount 3529 -> 0, slice length 15442 -> 11913
ForceExceptionCheck #19 DSPint 0x80 - cycles=0, downcount 13719 -> 0, slice length 15438 -> 1719
ForceExceptionCheck #20 DSPint 0x80 - cycles=0, downcount 8430 -> 0, slice length 15444 -> 7014
ForceExceptionCheck #21 DSPint 0x80 - cycles=0, downcount 3140 -> 0, slice length 15443 -> 12303
ForceExceptionCheck #22 DSPint 0x80 - cycles=0, downcount 13297 -> 0, slice length 15443 -> 2146
ForceExceptionCheck #23 DSPint 0x80 - cycles=0, downcount 8000 -> 0, slice length 15441 -> 7441
ForceExceptionCheck #24 DSPint 0x80 - cycles=0, downcount 2711 -> 0, slice length 15441 -> 12730
ForceExceptionCheck #25 DSPint 0x80 - cycles=0, downcount 12866 -> 0, slice length 15439 -> 2573
ForceExceptionCheck #26 DSPint 0x80 - cycles=0, downcount 7574 -> 0, slice length 15442 -> 7868
ForceExceptionCheck #27 DSPint 0x80 - cycles=0, downcount 2273 -> 0, slice length 15444 -> 13171
ForceExceptionCheck #28 DSPint 0x80 - cycles=0, downcount 12433 -> 0, slice length 15440 -> 3007
ForceExceptionCheck #29 DSPint 0x80 - cycles=0, downcount 7138 -> 0, slice length 15424 -> 8286
ForceExceptionCheck #30 DSPint 0x80 - cycles=0, downcount 1799 -> 0, slice length 15439 -> 13640
ForceExceptionCheck #31 DSPint 0x80 - cycles=0, downcount 11984 -> 0, slice length 15439 -> 3455
ForceExceptionCheck #32 DSPint 0x80 - cycles=0, downcount 6708 -> 0, slice length 15434 -> 8726
ForceExceptionCheck #33 DSPint 0x80 - cycles=0, downcount 1416 -> 0, slice length 15441 -> 14025
ForceExceptionCheck #34 DSPint 0x80 - cycles=0, downcount 11563 -> 0, slice length 15443 -> 3880
ForceExceptionCheck #35 DSPint 0x80 - cycles=0, downcount 6272 -> 0, slice length 15438 -> 9166
ForceExceptionCheck #36 DSPint 0x80 - cycles=0, downcount 951 -> 0, slice length 15438 -> 14487
ForceExceptionCheck #37 DSPint 0x80 - cycles=0, downcount 11135 -> 0, slice length 15442 -> 4307
ForceExceptionCheck #38 DSPint 0x80 - cycles=0, downcount 5843 -> 0, slice length 15405 -> 9562
ForceExceptionCheck #39 DSPint 0x80 - cycles=0, downcount 546 -> 0, slice length 15439 -> 14893
ForceExceptionCheck #40 DSPint 0x80 - cycles=0, downcount 10696 -> 0, slice length 15444 -> 4748
ForceExceptionCheck #41 DSPint 0x80 - cycles=0, downcount 5404 -> 0, slice length 15406 -> 10002
ForceExceptionCheck #42 DSPint 0x80 - cycles=0, downcount 115 -> 0, slice length 15442 -> 15327
// etc...
```

on HLE + Cached Interpreter:

```
ForceExceptionCheck #1 ARAMint 246 cycles, slice length 7722 -> 2500
ForceExceptionCheck #2 ARAMint 246 cycles, slice length 7720 -> 5916
ForceExceptionCheck #3 SyncGPUCallback 1000 cycles, slice length 7717 -> 7711
ForceExceptionCheck #4 SyncGPUCallback 1000 cycles, slice length 7140 -> 1101
ForceExceptionCheck #5 SyncGPUCallback 1000 cycles, slice length 6039 -> 1671
ForceExceptionCheck #6 SyncGPUCallback 1000 cycles, slice length 7719 -> 3725
ForceExceptionCheck #7 SyncGPUCallback 1000 cycles, slice length 15436 -> 11990
ForceExceptionCheck #8 ARAMint 246 cycles, slice length 6960 -> 4015
ForceExceptionCheck #9 ARAMint 246 cycles, slice length 2944 -> 250
ForceExceptionCheck #10 ARAMint 246 cycles, slice length 2694 -> 249
ForceExceptionCheck #11 ARAMint 246 cycles, slice length 2445 -> 249
ForceExceptionCheck #12 ARAMint 246 cycles, slice length 2196 -> 249
ForceExceptionCheck #13 ARAMint 246 cycles, slice length 1945 -> 333
ForceExceptionCheck #14 ARAMint 246 cycles, slice length 1610 -> 273
ForceExceptionCheck #15 AICallback 0 cycles, slice length 1335 -> 628
ForceExceptionCheck #16 DSPint 0 cycles, slice length 15443 -> 5419
ForceExceptionCheck #17 DSPint 200 cycles, slice length 10019 -> 1310
ForceExceptionCheck #18 DSPint 0 cycles, slice length 15438 -> 7266
ForceExceptionCheck #19 DSPint 0 cycles, slice length 15442 -> 6573
ForceExceptionCheck #20 DSPint 0 cycles, slice length 15442 -> 11864
ForceExceptionCheck #21 DSPint 0 cycles, slice length 15443 -> 1719
ForceExceptionCheck #22 DSPint 0 cycles, slice length 15438 -> 7000
ForceExceptionCheck #23 DSPint 0 cycles, slice length 15438 -> 12317
ForceExceptionCheck #24 DSPint 0 cycles, slice length 15443 -> 2146
ForceExceptionCheck #25 DSPint 0 cycles, slice length 15443 -> 7441
ForceExceptionCheck #26 DSPint 0 cycles, slice length 15442 -> 12751
ForceExceptionCheck #27 DSPint 0 cycles, slice length 15444 -> 2580
ForceExceptionCheck #28 DSPint 0 cycles, slice length 15444 -> 7882
```

```
ForceExceptionCheck #29 DSPint 0 cycles, slice length 15439 -> 13157
ForceExceptionCheck #30 DSPint 0 cycles, slice length 15441 -> 3007
ForceExceptionCheck #31 DSPint 0 cycles, slice length 15426 -> 8286
// etc...
```

and on LLE:

```
ForceExceptionCheck #1 ARAMint 0x00 - cycles=246, downcount 2456 -> 246, slice length 4661 -> 2451
ForceExceptionCheck #2 ARAMint 0x00 - cycles=246, downcount 1091 -> 246, slice length 6695 -> 5850
ForceExceptionCheck #3 SyncGPUCallback 0x3e8 - cycles=1000, downcount 1072 -> 1000, slice length 2156 -> 2084
ForceExceptionCheck #4 SyncGPUCallback 0x3e8 - cycles=1000, downcount 7102 -> 1000, slice length 7130 -> 1028
ForceExceptionCheck #5 SyncGPUCallback 0x3e8 - cycles=1000, downcount 5003 -> 1000, slice length 7721 -> 3718
ForceExceptionCheck #6 SyncGPUCallback 0x3e8 - cycles=1000, downcount 4425 -> 1000, slice length 8815 -> 5390
ForceExceptionCheck #7 ARAMint 0x00 - cycles=246, downcount 2231 -> 246, slice length 5966 -> 3981
ForceExceptionCheck #8 ARAMint 0x00 - cycles=246, downcount 1984 -> 246, slice length 1984 -> 246
ForceExceptionCheck #9 ARAMint 0x00 - cycles=246, downcount 1737 -> 246, slice length 1737 -> 246
ForceExceptionCheck #10 ARAMint 0x00 - cycles=246, downcount 1491 -> 246, slice length 1491 -> 246
ForceExceptionCheck #11 ARAMint 0x00 - cycles=246, downcount 1245 -> 246, slice length 1245 -> 246
ForceExceptionCheck #12 ARAMint 0x00 - cycles=246, downcount 910 -> 246, slice length 997 -> 333
ForceExceptionCheck #13 ARAMint 0x00 - cycles=246, downcount 635 -> 246, slice length 662 -> 273
ForceExceptionCheck #14 AICallback 0x00 - cycles=0, downcount 1282 -> 0, slice length 1495 -> 213
ForceExceptionCheck #15 DSPint 0x08 - cycles=200, downcount 2140 -> 200, slice length 3131 -> 1191
ForceExceptionCheck #16 SyncGPUCallback 0x3e8 - cycles=1000, downcount 2163 -> 1000, slice length 10332 -> 916
9
ForceExceptionCheck #17 SyncGPUCallback 0x3e8 - cycles=1000, downcount 3708 -> 1000, slice length 4117 -> 1409
ForceExceptionCheck #18 SyncGPUCallback 0x3e8 - cycles=1000, downcount 1744 -> 1000, slice length 2021 -> 1277
ForceExceptionCheck #19 SyncGPUCallback 0x3e8 - cycles=1000, downcount 7516 -> 1000, slice length 8100 -> 1584
ForceExceptionCheck #20 SyncGPUCallback 0x3e8 - cycles=1000, downcount 3429 -> 1000, slice length 3486 -> 1057
ForceExceptionCheck #21 SyncGPUCallback 0x3e8 - cycles=1000, downcount 8164 -> 1000, slice length 11800 -> 463
6
```

Of note is that DSP-LLE does not seem to generate any DSPint events with type INT_DSP.

Digging further into the INT_DSP events, I noted that they all seem to derive from AXUCode::SignalWorkEnd, which pushes a DSP_YIELD mail whenever a command list is handled (makes sense, Clone Wars is just running a command list over and over).

Disabling those mails breaks the game entirely, and changing the interrupt timer to a nonzero value in GenerateDSPInterruptFromDSPEmu doesn't seem to affect the game.

#### #4 - 05/08/2017 12:55 AM - ligfx

When Clone Wars hangs on the loading screen, it's because it's caught in an infinite loop at code address 0x8000a060:

```
0x8000a060 addi r26, r26, 1
0x8000a064 bl ->0x802BF2D0 (OSGetTick)
0x8000a068 lwz r3, -0x3B40 (r13)
0x8000a06c lwz r0, 0x0004 (r3)
0x8000a070 cmpwi r0, 1
0x8000a074 beq+ ->0x8000A060

OSGetTick:
0x802bf2d0 mftbl r3
0x802bf2d4 blr
```

where r13 = 0x8065dae0, r3 resolves to 0x8065a808, and the memory at that location is 0x00000001.

Manually skipping past the beq+ instruction makes the game continue running.

#### #5 - 05/08/2017 02:14 PM - hthh

it's not the most coherent explanation, but I'm pretty sure this is the issue:

We hang in SoundEngineGC::PrepLoad because __AXOutDspReady is always 1 ("READY") which is caused by HLE DSP being too fast. The loop is checking that it can run 50 times with the DSP being idle, before the DSP becomes busy. It relies on seeing the 0 ("BUSY") state to break out of the counter loop.

Audio data is double-buffered: as one frame is processed by the DSP, the previous frame is being flushed to the output.

New command lists are submitted to the DSP as triggered by the AID interrupt, which indicates more audio data is needed for output. The bug occurs because the DSP triggers INT_DSP with DSP_YIELD much too quickly. In fact, it triggers before the AID interrupt handler completes. Because of this, immediately upon "rfi" (return from interrupt), we will enter the __DSPHandler, which will call _AXDSPResumeCallback and set the state back to READY.

(The cached-interpreter is uneffected because it doesn't check for pending exceptions/interrupts on rfi.)

This prevents the code in SoundEngineGC::PrepLoad from ever observing the "BUSY" value. Even though it is momentarily written to __AXOutDspReady, it changes back to "READY" immediately.

The problem in Dolphin is that AXUCode::SignalWorkEnd is invoked immediately after processing. In the absence of a DSP thread, this makes the result pretty much immediate, and in the presence of a DSP thread it makes the result pretty unpredictable (?)

I suspect we should synchronously attempt to count the number of cycles taken by a command-list and schedule the DSP_INT based on that, rather than triggering it immediately.

A workaround (that other people seem to have noticed?) is to make GenerateDSPInterruptFromDSPEmu schedule the event 814 ticks in the future, rather than zero. This might be too slow for other very fast operations, although it's only ~135 DSP cycles, so I wouldn't be entirely surprised if it worked everywhere.

We could also attempt to estimate when to trigger it from the DSP thread, or pass a flag so that only mail from SignalWorkEnd gets delayed by 814 ticks (which is probably *more* accurate, but not really very accurate at all).

My reversed code: http://codepad.org/JQ8nAKDf

#### #6 - 05/08/2017 08:30 PM - ligfx

Wow, great write-up hthh! That sounds right basde on what I was looking at as well.

I hacked in your idea of scheduling SignalWorkEnd() some number of cycles after receiving the command list mail (in this case, DSP-LLE seems to wait about 163622 cycles, but to make it easy I went with one DSP update of ~400000 cycles) and the game boots. This is probably a good way to do this if we can figure out how long the wait interval is supposed to be across games/command lists.

#### #7 - 05/18/2017 01:39 AM - hthh

Thanks for sharing the DSP-LLE cycle count - it's been helpful to me making sense of the "**AXNextFrame" function, which seems to generate the command list, but also to update a variable called "**AXCommandListCycles" (0x8065A7F8). Looking in the Dolphin debugger, __AXCommandListCycles has the value 26621, which when converted from DSP cycles to PPC cycles is about 160,000 and suspiciously close to your LLE cycle count.

What's most interesting about the function is that it initialises the cycle count to 425 before adding any commands, so my guess is that the minimum (empty) command buffer would take ~425 DSP cycles to process, or about 2500 cycles. I propose that 2500 is a safe minimum, and would fix this issue, but also not break any (as yet unidentified) games that do very trivial/fast things with the DSP.

There are many more accurate options we could pursue, but I'm fairly sure this is strictly better than zero (whereas a cycle count like 50,000 *might* unexpectedly cause slowdowns in situations we haven't yet seen).

#### #8 - 05/18/2017 04:41 AM - ligfx

Incredible work hthh. I modified my previous hacked code to use that number and uploaded a PR: AX-HLE: delay sending interrupt when done processing command list

#### #9 - 05/23/2017 04:40 PM - JosJuice

*- Status changed from New to Fixed*

*- Fixed in set to 5.0-3860*

https://dolphin-emu.org/download/dev/3a2ec8c8a11e571a19c3bee676b5ab21f5aa79c9/

#### #10 - 06/24/2017 06:50 AM - JosJuice

*- Has duplicate Emulator Issues #10240: Star Wars: The Clone Wars can still crash between missions added*

#### #11 - 06/24/2017 07:53 AM - JosJuice

*- Has duplicate deleted (Emulator Issues #10240: Star Wars: The Clone Wars can still crash between missions)*