To: all the licensees
From: Yuji Takahashi, Executive Vice President of Business Affairs
Media: Mail
Title: Important notice on software development due to some hardware parts change

(Contents)

Thanks to all of you, we achieved the shipment of 1 mil consoles at the end of May. The sales condition in Japan is very positive. The reason of the success is that various attractive software titles from you have been released continuously and that PlayStation console is marketed at an attractive price. We highly appreciate your cooperation and would like to ask for your further support to PlayStation.

Now, the sales in US will be started on Sep 9th and European business will also be started in Sep. In each area, the number of licensees are already more than 100. The expectation from each of the market is also very good.

So, the very big demand of PlayStation is expected all over the world.
And, SCE is aware that it is very important to keep supplying the very big number of PlayStation console to each market.
As you know, recently semiconductor industry is suffering from shortage of memory chip because of worldwide PC demand increase, especially in US. We, SCE has been and is going to suffering from the shortage of the high-performanced chips for PlayStation console. In order to solve the problem, we have reviewed our current parts design for PlayStation and have re-designed some of LSIs.

Now, we have completed the development of the new LSIs and the production of PlayStation with the new parts will be started in the end of this year. That will be a running change.

We named the hardware with new parts **"Revision-C system"**.

We are aiming at full compatibility in the **"Revision-C system"**. However, it is reported that the very small/subtle difference in the parts specification might cause some very subtle difference in its software operation/gameplay, even though it seems it is very rare case.

When the software program meets the following spec., it is reported that the possibility of having slightly different gameplay in Revision-C system from gameplay in current system will be bigger.

(1) VSync() function is not properly called.
(2) MoveImage() for rectangular region of less than 32 dot width is often used.

We'd like to ask for your cooperation to check all of your already-released titles and already-submitted master disks. If you find any suspicious title, please contact our Account executive immediately. As soon as we get your report, we also check the mentioned title in detail.

As for under-development titles, please refer attached technical advise to keep full compatibility. If you think it's difficult for you to check your title by yourself because of tight schedule or if you feel any difficulty to find out any suspicious phenomenon, please do not hesitate to contact our account executives. The information is also available in BBS.

As for will-be developed titles, we'd like to do our best to provide the circumstance that developer can start development without being aware of hardware parts difference. For the purpose, we'd like to release a new library to suite the current situation.

Your kind cooperation and understanding to this matter would be highly appreciated.

=====End of Memo=====

=====2. Technical advise=====

Let us inform you some technical advise on further software development under the circumstance of releasing Revision-C system. Revision-C system is a hardware of which graphics chips is changed. The main purpose of releasing Revision-C system is to maintain stable memory supply. Basically it is designed under the concept with keeping full compatibility with current machine. However there is a small difference between Revision -C system and current models Revision A and B as follows:

By the way Revision A is a hardware marketed for Japanese market and Revision B is a hardware marketed for US/European market. There is no problem on compatibility between Revision A and B.

1) Semi-transparent drawing is faster.
A application which uses many transparent drawing runs faster.

2) It runs slower if vertically thin drawing or MoveImage() are frequently used.
It will be obviously observed with the thin rectangle region of less than 16 dot-width.

Fundamentally those two points above are the differences between current system and revision-C system. The reasons are due to the differences of memory system which is used as frame buffers.

Followings are the details of each point.

(1) The possibility which the problem occurs would increase if the existence of GPU-bottleneck (*1), the use of semi-transparency and the no proper control to frame rate are simultaneously occurred.

```
(Example)   while (1) {
                DrawSync(0);
                VSync(0);
                DrawOTag(ot);
            }
```

In the example above, the following problem will happen in case of GPU-bottleneck.

Semi-transparent drawing is faster
-> DrawSync(0) ends earlier.
-> Reaches VSync(0) earlier.
-> If the drawing is ended near the 1/30 (or 1/15 sec), VSync(0) returns one frame earlier.
-> Frame rate increase (opposite of frame rate decrease) occurs
-> Movement on screen seems to be quicker occasionally.

The rate of speeding up the drawing will depend on how many semi-transparent is used.

(*1) GPU-bottleneck is the status which the drawing by GPU is the slowest situation among all procedures while one frame is generated and displayed.

[Countermeasure]
Control frame rate precisely by using VSync(n) function.

For example, in the scene which more than 80% are running with 30 frames, please set the frame rate 1/30 sec fixed by using VSync(2) instead of VSync(0) to make sure.

Related to this, followings are about frame rate synchronization.

[About Frame Rate Synchronization]

In PlayStation, the display region on the frame buffers can be switched asynchronous to video frame rate (1/60 sec). However, if the display region is switched at the irregular rate which is not multiple of 1/60 sec, the switching is not done in the vertical retrace period and a phenomena which flicker can be seen will occur. This may misguide users that a application has some inferiority.

Therefore, in the normal process, the switch of the buffers should be synchronized with vertical synchronization (V-BLNK). In (A), the switch of buffers depends on the slower one either of display or drawing and the switch became asynchronous to V-BLNK. Therefore, if it is not intentionally targeting special effects, please execute VSync(0) to synchronize when switching buffers.

```
(A)                             (B)
while(1){                       while(1){
    ...                             .....
    DrawSync(0);                    DrawSync(0);
    swap_buffer();                  swap_buffer();
                                    VSync(0);
    DrawOTag(ot);                   DrawOTag(ot);
}                               }
```

But, when the switch is forced to synchronized to V-BLNK, the movement of objects will be awkward because the frame rate is frequently changed between 1/60 sec and 1/30 sec in case the transaction ends around 1/60 sec. This may cause users claim as well.

In this case, please fix the frame rate to 1/30 sec by using VSync(2).

```
while(1) {
        ....
        DrawSync(0);
        VSync(2);           /* set to 1/30 sec fixed rate */
        swap_buffer();
        DrawOTag(ot);
}
```

Thus, as far as possible, please keep the frame rate constant by using VSync(n).

But depends on an application, it is not better to set the worst frame rate. Even in this case, the clock of program internal should not count the buffer switching, instead should use the absolute counter such as VSync(-1), RCnt3, etc.

```
(A)                             (B)
    while(1){                   while(1){
        DrawSync(0);                    DrawSync(0);
        swap_buffer();                  swap_buffer();
        Vsync(0);                       VSync(0);
        frame++;                frame = VSync(-1);
        DrawOTag(ot);                   DrawOTag(ot);
```

} }

By the means of frame counter counting shown in (B), the internal counter will not delay if the frame rate decrease momentarily due to the calculation or drawing overflow. If the counter is used to update the position of objects, the movement of objects can be kept naturally even if the frame rate drops.

(2) MoveImage() is slower than current system when it is executed to the thin rectangle region of less than 32 dot-width. For example, it would be the problem when the MoveImage() is frequently used to the thin rectancle region such as 8 by 240.

[Countermeasure]

- By using ResetGraph(1) just after VSync(0) to stop MoveImage(), the influence of slowness should confined within the frame, and avoid affecting other frames.

- In case of revision-C, the narrower the width of rectangle region is, the more prominent the difference from the current system is. Therefore, please make the width of the rectangle region wider.

Two approached above can be considered, but the former would be better.

It need to be more careful if the interace mode is used with this function. Even if the frame rate is slower than 1/60 sec, the screen will be disordered. For example, when the drawing is done with the interlace mode (640x480 etc) and single buffer, please use ResetGraph(1) instead of DrawSync(0).

For your reference, following chart show how large MoveImage() in revision-C slower when compared to current system.

WIDTH	Revision-C / Revision-A (%)
32	100 %
16	85 %
8	77 %
4	70 %