GDI/DirectDraw Accelerator Integration on WINCE5 and WINCE6/7

1. Integration with the current display driver

For easily adding/removing the GDI/DirectDraw accelerator, a compiler switch USE_GC2D_ACCELERATOR is used to control it. It should be defined if you want to enable hardware acceleration.

Supposing the user's GDI driver class is CSample, which may be derived form GPE or DDGPE, several places need to be changed in CSample implement code.

1.1. CSample must be derived from class GC2D_Accelerator, the definition of class CSample must be changed to:

```
class CSample :
#ifdef USE_GC2D_ACCELERATOR
        public GC2D_Accelerator
#else
        public GPE        // or DDGPE
#endif
```

a.) gc_gpe_accelerator.hpp should be included in CSample definition header file. A reference code is like:

```
#ifdef USE_GC2D_ACCELERATOR
#include <gc_gpe_accelerator.hpp>
#endif
```

b.) The class GC2D_Accelerator can be derived from GPE or DDGPE depending on whether the environment variable DERIVED_FROM_GPE is set to 1. (Instead, you can add "DERIVED_FROM_GPE=1" in the file sources.cmn in the directory gchal)

If GC2D_Accelerator is derived from GPE, link gc2d_GPE_accelerator.lib and the macro DERIVED_FROM_GPE should be defined in the file SOURCES of the display driver. If derived from DDGPE, link gc2d_DDGPE_accelerator.lib. The following is a reference which could be added to the end of the SOURCE file of the display driver.

```
!IF "$(DERIVED_FROM_GPE)" == "1"
DERIVED_BASE_CLASS = GPE
CDEFINES=$(CDEFINES) -DUSE_GC2D_ACCELERATOR -DDERIVED_FROM_GPE
!ELSE
```

```
DERIVED_BASE_CLASS = DDGPE
CDEFINES=$(CDEFINES) -DUSE_GC2D_ACCELERATOR
!ENDIF

SOURCELIBS= \
        $(SOURCELIBS) \
        $(_TARGETPLATROOT)\lib\$(_CPUINDPATH)\GC2D_$(DERIVED_BASE_CLASS)_Accelerator.lib \
        $(_TARGETPLATROOT)\lib\$(_CPUINDPATH)\k_libGAL.lib

INCLUDES= \
        $(INCLUDES); \
        $(_WINCEROOT)\public\gchal\driver\GDI\GC2D_Accelerator; \
        $(_WINCEROOT)\public\gchal\hal\inc
```

1.2 Accelerator Initialization and Deinitialization

GC2D_Accelerator::GC2DInitialize() member function should be called before GDI/DirectDraw acceleration, it can add to CSample Constructor. GC2D_Accelerator::GC2DDeinitialize() can add to CSample destructor. A reference code is like:

```
CSample::CSample()
{
        //user private constructor code
        ……

#ifdef USE_GC2D_ACCELERATOR
        if (GC2DInitialize())
        {
                RETAILMSG(TRUE, (TEXT("GC2D: accelerator initialization succeed (enable=%d, sync mode=%d).\n"), GC2DSetEnable(), GC2DChangeSyncMode()));
        }
        else
        {
                RETAILMSG(TRUE, (TEXT("GC2D: GDI accelerator initialization failed!\n")));
```

```
        }
#endif
}

CSample::~ CSample ()
{
#ifdef USE_GC2D_ACCELERATOR
        GC2DDeinitialize();
#endif
        //user private destructor code
        ……
}
```

1.3 GDI/DirectDraw accelerator support

To enable GDI/DirectDraw accelerator, GC2D_Accelerator::GC2DBltPrepare() and GC2D_Accelerator:: GC2DBltComplete() should be added into CSample::BltPrepare() and CSample::BltComplete()by separately. A reference code is like:

```
SCODE CSample::BltPrepare(GPEBltParms *pBltParms)
{
#ifdef USE_GC2D_ACCELERATOR
        if (GC2DIsValid())
        {
                if (S_OK == GC2DBltPrepare(pBltParms))
                        return S_OK;
        }
#endif //USE_GC2D_ACCELERATOR

        //user private code
        ……
        return S_OK;
}
```

```
SCODE CSample::BltComplete(GPEBltParms *pBltParms)
{
#ifdef USE_GC2D_ACCELERATOR
        if (GC2DIsValid())
        {
                GC2DBltComplete(pBltParms);
        }
#endif //USE_GC2D_ACCELERATOR

        //user private code
        ……
        return S_OK;
}
```

1.4 hardware status controls

After adding the GDI/DirectDraw accelerator, the hardware status can be controlled in the functions CSample::IsBusy() , CSample::WaitForNotBusy() and CSample:: PowerHandler(BOOL). The implement code should be changed like:

```
VOID CSample::WaitForNotBusy()
{
#ifdef USE_GC2D_ACCELERATOR
        if (GC2DIsValid())
        {
                GC2DWaitForNotBusy();
        }
#endif //USE_GC2D_ACCELERATOR

        //user private code
        ……
        return;
```

```
}
int CSample::IsBusy()
{
	int ret = 0;
#ifdef USE_GC2D_ACCELERATOR
	if (GC2DIsValid())
	{
		ret = GC2DIsBusy();
	}
#endif //USE_GC2D_ACCELERATOR

	//user private code
	……
	return ret;
}
int CSample:: PowerHandler(BOOL bOff)
{
	int ret = 0;
#ifdef USE_GC2D_ACCELERATOR
	if (GC2DIsValid())
	{
		ret = GC2DpowerHandler(bOff);
	}
#endif //USE_GC2D_ACCELERATOR

	//user private code
	……
	return ret;
}
```

2. Build

2.1 make sure the GPU driver (libgalcore.dll) and the GAL library (libgal.dll for WINCE500; k_libgal.dll for WINCE600 and above) have been built successfully;

2.2 build the GDI/DirectDraw accelerator library in gchal/driver/GDI -- gc2d_xxGPE_accelerator.lib;

2.3 build the integrated display driver linking libgal.lib/k_libgal.lib and gc2d_xxGPE_accelerator.lib.

3. Configuration in the registry file

There are three keys for the accelerator config in the file gchal.reg:

A.)GC2DAcceleratorEnable: enable different features. If Bit0 is set to 0, the accelerator and the other bits will be disabled.

Bit0: BLTPREPARE

Bit1: ALPHABLEND

Bit2: TRANSPARENT

Bit3: STRETCH

Bit4: MASK

B.)GC2DAcceleratorSyncMode: set different sync mode;

0: force mode

1: async mode

2: full async mode

C.)GC2DAcceleratorMinSize: set the minimum pixel number for hardware to render;

4. Running & Validation

4.1 After you download your image to the device, you may get the info from the output of Windows CE Debug window, for example:

GC2D: GDI accelerator initialization succeed (enable=0x1f, sync mode=1).

4.2 Run the GDI test in CETK/LTK to validate the accelerator.