T E C H N O T E :
Power Management & Servers: Auto Restart From Power Failure

By Vinnie Moscaritolo
vinnie@apple.com
<http://webstuff.apple.com/~vinnie/>
Apple Developer Technical Support (DTS)

Automatically restarting from a power failure is an important feature for any server. Normally, powering up a Macintosh computer requires manual intervention by the user. It is possible, however, to configure the Macintosh hardware, so that the system will power itself up anytime primary A.C. is available.

This Note discusses how to communicate with the Macintosh's internal power management microcontroller and is important for developers who design software that must run in an environment where little or no human intervention is available.

The Process of Turning On Your Macintosh

Most server software is designed to run in an environment where you're not guaranteed human supervision. A remote server, for example, ought to be able to restart itself after a power failure without any manual interaction.

By default, Macintosh computers are designed to be used in a desktop environment, where the user must power up by manually pressing the power-on key. However, in most cases it is possible to configure the Macintosh firmware, so that the system will power itself up the next time that primary A.C. is available.

Most Macintosh computers are equipped with an internal microcontroller that, among other things, manages the Macintosh power providing **Soft Power Control**. This microcontroller, a custom ASIC designed specifically for Apple, dictates the circumstances under which the Macintosh will initiate a power up cycle. Typical power-on options include:

- Manual depression of the Power-key on the keyboard.
- Manual depression of a momentary contact switch, usually mounted on the rear chassis of the system.
- External power up, such as occurs when a NuBus card asserts the `PFW` signal directly.
- The internal power up alarm specified by the Power Manager command `SetStartupTimer` becomes active.
- On Macintosh units that support this feature, there is a Wakeup line connected to the serial ports GPI pin. A typical application of this feature is to power up the Macintosh when it is used as an answering machine or modem server.
- When the system is configured in **Server Mode** and A.C. power is restored.

This Note only concentrates on the last two options, Server Mode and Wakeup Mode, and how to enable/disable them.

Introducing The Cuda Manager

Server Mode is enabled by accessing an internal piece of Macintosh system software known as the **Cuda Manager** (also known as Egret). The Cuda is the firmware that communicates to the microcontroller responsible for managing the Macintosh power.

Keep in mind that not all Macintosh models have the Cuda Manager or **Soft Power Control**. Soft Power Control exists only on systems where A.C. voltage is always available to the power supply, and the power supply is controlled by the state of the power fail warning (PFW) signal.

On the other hand, **Passive Power Control** exists in systems where the power supply is turned off or on by a switch directly in line with the primary A.C. voltage to the supply.

For example, Macintosh Classic and Macintosh II LC employ Passive power control, while the Macintosh II and most Power Macs use Soft Power.

There are also certain Macintosh models, such as the Color Classic and LC475, LC575 CPUs, that implement a **Pseudo Soft Power** supply control. In those cases, the keyboard power key can be used to initiate a power up of the system, but the chassis switch is wired directly to the power supply and is not utilized by Cuda.

Determining If Your Mac Supports the Cuda

The proper way to determine if a particular Macintosh model supports the Cuda Manager is as follows:

1. Use the Gestalt function to check for the `gestaltHardwareAttr` 'hdwr' selector. Then check the response for `gestaltHasSoftPowerOff` (bit 19). This will indicate if the Macintosh supports Soft Power

2. Use the `GetOSTrapAddress` to check for the existence of the Cuda Manager dispatch trap known as `_EgretDispatch` ($A092)

```
long unknownTrapAddr;
unknownTrapAddr = GetOSTrapAddress( _Unimplemented );
if( unknownTrapAddr == GetOSTrapAddress( _EgretDispatch ) )
```

3. In addition to checking the `_EgretDispatch` trap vector, you also need to ensure that the Cuda Manager software is loaded. There are a few Macintosh ROMs that, even though they implement the Cuda trap, do not have appropriate Cuda hardware. On these machines, invoking the `_EgretDispatch` trap call *will result in a bus error*.

   In order to verify that the Cuda software was loaded, you need to check that the internal low memory global at location `0xDE0` does not equal -1.

```
#define CudaBase        0x00000DE0
typedef Ptr             *CudaGlobalsPtr;

Boolean CheckForCuda( void )
{
        long    unknownTrapAddr;

        unknownTrapAddr  = GetOSTrapAddress( _Unimplemented )
        if( unknownTrapAddr == GetOSTrapAddress( _EgretDispatch ) )
                return FALSE;
        if( CudaBase == (CudaGlobalsPtr) -1 )
                return FALSE;
        return TRUE;
}
```

Calling the Cuda Manager

Only after you have established the existence of the Cuda Manager should you then pass commands to it. You can do this by issuing a trap dispatch call to the Cuda Manager through the `$A092` trap.

On entry, register A0 must contain a pointer to a parameter block which describes the type of function to be executed by Cuda and how a response to execution of that function is to be returned.

When issuing a Cuda dispatch trap from C, it is necessary to pass a pointer to the Cuda parameter block to the trap using register A0 as a pointer to the parameter block. The following function prototype can be used to interface to the Cuda Manager:

```
#pragma parameter Cuda( __A0 )
void Cuda( CudaPB* myPB ) = 0xA092;
```

Cuda Manager Parameter Block Structure

The Cuda Manager parameter block structure used by all the Cuda Management functions is defined as follows:

```
#define  pseudoPkt      0x01    /* Cuda Pseudo Packet         */
#define  EnDisFileS     0x13    /* enable/disable file server flag    */
#define  WakeupMode     0x23    /* Enable/Disable WakeUp Mode */
#define  GetPwrFailTime 0x27    /* read time of last power failure    */

typedef struct {

      unsigned char         pbCmdType;  // Command Type,  always  'pseudoPkt'
      unsigned char         pbCmd;      // Command
      union{                            // parameter to pass

              unsigned char pByte[4];
              unsigned short            pWord[2];
              unsigned long pLong;

                            }pbParam;
      unsigned short        pbByteCnt;  // Number of bytes passed in buffer
      unsigned char         *pbBufPtr;  // Pointer to a buffer.
      unsigned char         pbFlags;    // Flags returned by Cuda
      unsigned char         pbSpare;    // reserved
              short         pbResult;   // Result code returned by Cuda
      ProcPtr               pbCompletion;      // Routine to be called on
completion

}CudaPB, *CudaPbPtr;
```

The Cuda Manager functions that you need to use are classified as Pseudo Device functions. Typically, parameters of four bytes or less can be passed to or from the Cuda in the parameter block. An error code may also be returned in the event of an unsuccessful completion of a function.

Enable / Disable File Server Mode

This call is used to inform Cuda whether the system should be configured as a standard personal computer or as a file server. When configured as a file server, the system will power itself up any time primary A.C. is available. When configured as a standard personal computer, the system must be powered up either by manual intervention of the user or the power up timer. A value of zero passed to pbParam will configure the system as a standard personal computer while a non zero value will configure the system as a file server.

```
short Enable_FS_Mode()
{
 CudaPB                thePB;

 if(!CheckForCuda() ) return( kNoCudaError);

 thePB.pbCmdType       = pseudoPkt;
 thePB.pbCmd           = EnDisFiles;
 thePB.pbParam.pByte[0]= 1;    //enable or disable
 thePB.pbParam.pByte[1]= 0;
 thePB.pbParam.pByte[2]= 0;
 thePB.pbParam.pByte[3]= 0;
 thePB.pbByteCnt       = 0
 thePB.pbBufPtr        = nil;
 thePB.pbResult        = 0;
 thePB.pbCompletion    = nil;

 Cuda( &thePB );
 return  thePB.pbResult
}
```

Note: The mode will revert to a standard personal computer if the Shutdown command is issued.

Working with a UPS

File Server Mode can be used in conjunction with an external Uninterruptible Power Supply (UPS). But it requires that you programatically shut down the Macintosh before the UPS powers off. Since the File Server Mode state is only valid until the next shutdown, you must “force” the shutdown into not reverting the Cuda into personal computer mode.

One way to accomplish this is to interact with the shutdown process, as follows:

Near the end of the shutdown process the ShutDownMgr will check the `gestaltHardwareAttr` 'hdwr' selector's response for `gestaltHasSoftPowerOff` (bit 19). If this machine does not have soft power, it puts up the *Safe to Shut Off Your Computer* alert.

As part of your UPS Manager's setup, you can use the `ShutDwnInstall` procedure to install a custom shutdown proc that will run before the computer powers down. For example:

```
ErrNo = ShutDwnInstall (myShutdownProc, sdOnPowerOff);
```

Then in your Shutdown proc, you can use the `ReplaceGestalt` function to intercept the ShutDownMgr's Gestalt lookup, so that it returns a response that has the `gestaltHasSoftPowerOff` bit reset.

Note: Be sure that you read the "Shutdown Manager" chapter of *Inside Macintosh: Processes* to understand exactly what happens in the shutdown process and the correct way to shut down a Macintosh computer.

Power Fail Time Clock

The `GetPwrFailTime` call is used to read the 32 bit integer that represents the last time the system was powered down. The Cuda maintains this in the number of elapsed seconds since January 1, 1904.

```
Word Get_PwrFail_Time()
{
myCudaPB.pbCmdType      = pseudoPkt;
myCudaPB.pbCmd          = GetPwrFailTime;  //  ($27)
myCudaPB.pbParam.pLong  = Time;
myCudaPB.pbResult       = noErr;
myCudaPB.pbCompletion   = NIL;

Cuda( &myCudaPB );
return myCudaPB.pbResult;
 }
```

Note: This function is only available in Cuda Manager 3.0 or greater. The Cuda Manager can be tested by checking the 'cuda' Gestalt selector.

Summary

Although this Technote describes how to configure your Macintosh to automatically restart from a power failure, not all Macintoshes support this feature. This is because there are other ways for Macintosh computers to implement soft power control. For instance, PowerBooks utiltize a completely different system. This is why your code should follow the steps outlined in this Note to check if the Cuda/Egret trap is implemented.

Since accessing the Cuda Manager requires intimate knowledge of low memory globals, it may not work in later Mac OS releases.

Further References

- *Inside Macintosh: Devices, Chapter 6, Power Manager Reference*
- Technote 1046: *Inside Macintosh: Devices, Power Manager* Addenda

Acknowledgments

Special thanks to Ray Montagne, Tom Maremaa, and Mark Baumwell.