HOWTO for Powersave ADNP/1520

1. Why need Powersave and how it works?

DOS-Applications stays mostly in an idle loop and waits for keys or other events. The CPU can go into HALT-state to save the power. In HALT-State mostly of internally clocks are still halted. The CPU leave the HALT from any hardware interrupts, for sample Timer (IRQ0), serial input (IRQ4) or keyboard (IRQ3 for TRM series), LAN (IRQ5 or IRQ7), or all the others that are enabled.

Without save power by software the CPU needs coolers, head pipe or head sink.

Summary:
The CPU must not wait with full speed. Lets sleep it in HALT and wakeup on every interrupt event.

2. How can bring CPU in the HALT state?

Add a Assembler function "hlt" in the inner loop of all your waits.
Install a TSR that does bring CPU in HALT, if your application BIOS calls for "waiting key".

2.1. How can modify my sources to save the power with HALT?

Typically you have such abstracted Program-Loop where you waits for events:

```
Loop:
        if key_pressed
        then
                Call Key_handler
        endif

        if LAN_messsage_received
        then
                Call Do_any_with_LAN
        endif
Goto Loop
```

Change this by adding the HALT command:

```
Loop:
        Set Idle_flag = TRUE

        if key_pressed
        then
                Call Key_handler
                Set Idle_flag = FALSE
        endif

        if LAN_messsage_received
        then
                Call Do_any_with_LAN
                Set Idle_flag = FALSE
        endif

        if Idle_flag
        then
                asm "hlt"
        endif
Goto Loop
```

2.2. Example for a Event-Waiting-Loop as C-Source

```
while (1) {
        int idle = 1;                   // Assume, we are idle

        if (_bios_keybrd(1)) {          // Any Key Pressed?
                int key = _bios_keybrd(0); // Read the key from DOS
                HandleMyKey(key);       // call a function to do something
                idle = 0;               // Don't HALT in this Loop
        }

        if (Lan_Packet_ready()) {       // Check the LAN
                HandleMyLANPacket();    // Handle the LAN message
                idle = 0;               // Don't HALT in this Loop
        }

        if (idle) {                     // Are we idle?
                asm {
                        hlt             // Yes: HALT now and waiting for IRQ
                }
        }
}
```

2.3. How can wait with better power save (delay-function)?

Abstract programming sequence for a power saved delay loop:

```
MyDelay:
        set EndTime = CurrentTime + TimeDiff
DelayLoop:
        if CurrentTime >= EndTime
        then
                exit
        endif

        Asm "hlt"
Goto DelayLoop
```

2.4. Example for a Delay-Loop as C-Source

```
void MyDelay (int milliseconds)
{
        long end, current;

        current = _bios_timeofday();          // Get current time
        end = current + milliseconds / 54;    // Setup exit condition

        while (current < end) {                       // exited the exit time?
                asm {
                        "hlt"                 // Wait for Timer and other IRQs
                }

                current = _bios_timeofday();  // Get current time
        };
}
```

3. Have no sources, or can not modify it. Can install a driver?

3.1. KEY_IDLE.COM - Goes into idle in the waiting for keys

This driver is the best, to save mostly of power. This driver goes into HLT, if no keys are in keybuffer and a user application calls the INT 16H/AH=00h or AH=10h (read key). The driver goes also into HALT, if DOS calls the Idle-Interrupt 28H longer than 200ms. The HALT will be leave from any hardware interrupts. It's reenter the idle state, if only the timer (IRQ0) was the wakeup. For other wakeup events, the idle state enters again after a time of round about 200ms.

The first call of KEY_IDLE.COM installs it as TSR. The TSR can uninstall with a second call or force it with "KEY_IDLE.COM /U".

This driver must use, if applications calls the BIOS-Function 16H/AH=00h or 16H/AH=10h. This driver is usable with applications, that calls kbhit and getch.

3.2. I28_IDLE.COM - Goes into idle from INT 28H DOS idle call

This driver goes into HALT, if DOS calls the Idle-Interrupt 28H. This is typically for C-Function kbhit and getch and the DOS-Prompt self.

The first call of I28_IDLE.COM installs it as TSR. The TSR can uninstall with a second call or force it with "I28_IDLE.COM /U".

This driver can use, if applications calls with kbhit and getch.
It's not usable for applications, that calls BIOS INT 16H for getting keys.

3.3. CPU100.COM - Set CPU speed to 100 MHz

This is not a real power save. It sets the CPU clock to 100 MHz.
The CPU needs a cooler under DOS, if you don't use HALT in your application.

4. FAQ

======

1.) Why need Powersave?

The CPU works as an embedded system. If you put it into a case must doing something to save the totally power consumption. Or must build in a heat dissipation. With a software power-save, you can use the CPU in a case without coolers.

2.) Have no interrupts in my waiting loop. Can I call asm "hlt"?

Yes. In every case, you are idle, you can call the asm "hlt". Your application will wakeup by timer interrupt every 54ms. This should be good for low power idle waits.

3.) What is the difference between kbhit/getch/bios_keybrd?

kbhit and getch are DOS functions. DOS will call INT 28h, if no key is in buffer. If you use these functions, the I28_IDLE.COM will do enough for saving power.

bios_keybrd is a call software interrupt 16h. The subfunction AH=01h is a "check key ready", the subfunction AH=00h is going into "wait for key". The KEY_IDLE.COM detects the function "wait for key" (AH=00h, or AH=10h) and goes into HALT, if no key is in buffer.

Warning: The TSR can not save power, if an applications is calling only INT 16h/AH=01h (check key ready).

4.) What is the best way?

You should know your application and insert an asm-instruction "hlt" in your loop for idle state.

If you don't know your software internals, or can not change it, try the KEY_IDLE and than I28_IDLE.

5.) Are some negative side effects known?

Some software is running slower, if they calls to often the kbhit (more as ones per loop). Try to use the KEY_IDLE.COM, this goes only into HALT, if no keys are in the keyboard buffer.

With networking you have an other problem: Broadcast will wakeup from idle state. After a networking packet the TRS leave the idle state and goes into idle only after a delay of some millisecond.

You will have no negative effects, if your application use the asm "hlt" in waiting loops and you not use the TSR.

6.) How can check the working?

Check the input current for the system. If power save is working you will see typically 60% of normal power consumption.

7.) Power save is no working in delays?

That's true. Standard delay functions are simple multiple loops in loops. You CPU is waiting for nothing with full speed. Rewrite it with a time based loop and call an asm "hlt" or kbhit in the inner loop.