

*The secrets to unleashing the full power  
of NeoBook Professional for Windows  
– at your fingertips!*



# **DEVELOPING A MULTIMEDIA KIOSK**

With NeoBook Professional  
for Windows Software

**Ronnie Toon**

## About the author

I'm currently working in a multinational multimedia development company headquartered in Singapore. My computer interest includes toying with graphics and multimedia software like **Adobe PhotoShop™**, **Adobe Acrobat Exchange™**, and **NeoBook® Professional for DOS™ / Windows™**. Many years ago (*during XT/AT/386 days*), I have developed customized programs many years ago, in QuickBasic, FromBat (Batch compiler), RSD GameMaker, and Objects Layout, a true CASE and OOPS software. I'd considered myself a novice programmer. I also love playing mind-boggling computer puzzle games that makes me think and 'exercise' my brain! Right now, my free time is usually spent on family and surfing (what else!) the Internet, searching for exciting "Shareware" and "Freeware" graphics/multimedia applications and utilities... and games.

## Acknowledgement

My inspiration for writing this article came mainly from the success of one of the first NeoBook site "*Personal NeoBook Journal*" setup by John Eric Arterberry. Because of the success in information-sharing and active user-participation at John's site, I have gathered some tips from fellow NeoBook users.

Special thanks also go to Neosoft Corporation (*developers of NeoBook® and other wonderful software titles*), for developing such a wonderful multimedia authoring software.

Lastly, the time and effort to made the first step and start actual writing would not have been made possible without the support of my family, who have to bear with my constant late night's sleep.

## Contacts:

### Ronnie Toon

Email: [toonbc@singnet.com.sg](mailto:toonbc@singnet.com.sg)

Home Page: <http://www.singnet.com.sg/~toonbc>

### NeoSoft Corporation

Email: [info@neosoftware.com](mailto:info@neosoftware.com)

Web Site: <http://www.neosoftware.com>

### John Eric Arterberry

Email: [arterberry@geocities.com](mailto:arterberry@geocities.com)

Home Page: <http://www.geocities.com/SiliconValley/Peaks/2461>

## *Disclaimer*

THE USER OF THE INFORMATION PROVIDED IN THIS ARTICLE ASSUMES ALL RISK OF ITS ACCURACY AND FOR ITS USE. THIS ARTICLE IS BEING PROVIDED "AS-IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. ALL OTHER LIMITATIONS ON LIABILITY CONTAINED IN THE APPLICABLE SOFTWARE PRODUCT END USER LICENSE AGREEMENT SHALL APPLY.

I, RONNIE TOON ("AUTHOR") ASSUMES NO RESPONSIBILITY FOR ERRORS OR OMISSIONS IN THIS ARTICLE. THIS ARTICLE MAY INCLUDE TECHNICAL OR OTHER INACCURACIES OR TYPOGRAPHICAL ERRORS, AND CHANGES MAY BE PERIODICALLY ADDED TO THE INFORMATION HEREIN.

THE AUTHOR DOES NOT GUARANTEE THAT SOLUTIONS SUGGESTED IN THIS ARTICLE WILL BE EFFECTIVE IN THE USER'S PARTICULAR SITUATION. IF THE USER IS NOT FAMILIAR WITH ANY OF THE STEPS LISTED IN THE SOLUTION, THE AUTHOR ADVISES THAT THE USER DOES NOT PROCEED WITHOUT FIRST CONSULTING ADDITIONAL RESOURCES.

Revision 1.0 (15<sup>th</sup> March 1998)

Revision 1.1 (12<sup>th</sup> April 1998)

## **What Bill Gates says about 'Internet' Kiosks:**

*From the book:*

### **Bill Gates "The Road Ahead" 1996**

*"...you'll still have access to the (Information) highway by using kiosks - some free, some requiring payment of a fee - which will be found in office buildings, shopping malls, and airports in much the same spirit as drinking fountains, rest rooms, and pay phones. In fact, they will replace not only pay phones but also banking machines, because they will offer their capabilities as well as all the other highway applications, from sending and receiving messages to scanning maps and buying tickets. Access to kiosks will be essential, and available everywhere. Some kiosks will display advertising links to specific services when you first log on - bit like the phones in airports that connect right to hotel and rental-car reservations."*

© William H Gates III

**INTRODUCTION.....6**

**SECTION 1: GETTING STARTED .....7**

- 1. IMPRESSIVE SCREENS .....7
- 2. VOICE-OVER NARRATION AND BACKGROUND MUSIC .....8
  - 2.1 WAVE files.....8
  - 2.2 MIDI files .....9
- 3. INTERACTIVE, CLICK-ABLE ‘PUSH’ BUTTONS .....9
- 4. SELF-RUNNING SLIDE-SHOW AFTER A CERTAIN PERIOD OF INACTIVITY ....10
- 5. INDIVIDUAL TIME-EVENT TO ‘RETURN’ TO SPECIFIC PAGE UPON COMPLETION OF A TASK. ....10
- 6. DISPLAYING OF VIDEO CLIPS WITHIN PRE-DEFINE WINDOW AREA AND/OR OPTION TO MAXIMIZE IT TO A FULL SCREEN .....10
- 7. ANIMATIONS (AUTOMATIC OR MANUAL).....10

**SECTION 2: IDLE EVENT.....11**

- CREATING INDIVIDUAL IDLE TIME EVENT.....12
  - For NeoBook v3.1c only*.....16
  - For NeoBook v3.2* .....21

**SECTION 3 – USING ACTION EVENT COMMANDS.....24**

- MCICOMMAND.....24

**TIPS & TRICKS .....28**

- 1. SETTING UP KEYBOARD CONTROLS.....28
- 2. RUNNING PUBLICATION AT SPECIFIED SCREEN MODE .....29
- 3. INCLUDING FONTS IN YOUR COMPILED PUBLICATION.....30

**APPENDIX 1 .....31**

- SPECIFYING INDIVIDUAL TIME EVENT WITH NBW V3.1C.....31

**APPENDIX 2 .....34**

- SPECIFYING INDIVIDUAL TIME EVENT WITH NBW V3.2.....34

**FREQUENTLY-ASKED QUESTIONS .....36**

### *Introduction*

While developing a multimedia kiosk demo recently with NBW v3.1c, there were some “ideal features” that I’d hope to incorporate into my demo but couldn’t simply because these features are not yet available in NeoBook® Professional for Windows (NBW). “*Perseverance leads to success*” - and I’ve come up with a workaround solution to ‘emulate’ these features into my demo.

As many of you are aware of, there are so many books, ‘*dummy guides*’, and online publications educating users on using popular software like **Adobe PhotoShop™** and **Microsoft PowerPoint™**. And the only book on NeoBook that I’ve come across is the ‘*User Guide*’ that came bundled with the registered software. Because of the lack of supporting documentation on NBW, I have decided to “*start the ball rolling*” by planning to write a series of articles based on the “**By Example...**” concept. By sharing my experiences, tips and techniques (*that I’ve learned the hard way*) in getting the most out of the NeoBook software, I hope you need not ‘re-invent’ the wheel should you need to use one of more of these features in your next publication.

This is my first article on NBW. It will not teach you how to use the software (*read the User Guide for NBW if you need help*). Instead, it’ll basically focus on how you can “*get the most out of using*” the NBW software. I hope you’ll enjoy reading this article and (*hopefully*) gather some tips along the way. And if you do, tell me. Do expect more articles coming your way, if you first made an effort to feedback me with your comments on what you think about this article (*be they negative, positive, or constructive*), suggestions and even tricks to enhance the techniques that I’ve used.

If you have happens to have an interesting idea, or would like to emulate certain features that you’ve seen in some other demos but do not know how to do so with NBW, write to me. If I know it it can be done, (*who knows*) my next article may address that. If not, your request will also be made available on my Home Page where fellow NeoBook users can share a tip or two.

## Section 1: Getting Started

Playing off the proverbial "a picture is worth a thousand words," an interactive visual display (kiosk) using computer-based multimedia could be the ideal marketing tool needed to transform your target market into well informed and motivated customers.

*A typical kiosk demo will usually have the following features:*

1. Impressive screens;
2. Voice-over narration and background music;
3. Interactive, click-able 'push' buttons (with sound) that leads you to any other screens, in non-linear way;
4. Self-running slide-show when there is no user-intervention or mouse activity after a certain period of time;
5. Individual time-event to 'return' to specific page upon completion of a task.
6. Displaying of video clips (AVI, MOV, MPG, VOB) within pre-define window area and/or option to maximize it to a full screen; and
7. Animations (automatic or manual).
8. Password-controlled or hidden 'hotkey' to prevent the demo program from exiting accidentally.

Yes, it is true that many of the kiosk demos were developed using the more common and popular multimedia development software like **Macromedia Director™**, **Authorware™** and **Asymetrix Toolbook™**. Although NBW may lack certain powerful but essential features usually available in other competitor's software, you can also use it to develop kiosk demo easily, apart from the other multimedia publications like E-Zines, E-Books, and slideshows that NBW have been known for. Read further, and allow me to show you how I did it.

Firstly, there are some factors that you may want to consider first when developing a kiosk demo; regardless of whether you are an experienced multimedia developer or not. Some of the features available in NBW are also available in the other multimedia development software mentioned earlier. Although some of the implementations may be different, the results are similar. Likewise, there are features found in these other software but missing in NBW. Each software has its advantages, features; and of course, limitations. By fully understanding the limitations, applying it in your publications, you can transform it as one of the features in your publication!

### 1. Impressive screens

Creating multimedia kiosk-like demo is not similar to designing 'static' layout for packaging, advertisements, and traditional books. Traditional mediums are displayed in linear way, while multimedia screens are designed with a non-linear

way; where hyperlinks are available to jump to one page or another, in any way possible. The keynote here is 'inter-activity' in all screens, and at any one screen, there should be at least an option to return to a previous page, sub-menu or main menu. This means that on each screen page, the interactive 'buttons' should be considered as a graphic element in the designing process. For example, if you are reading this document through Adobe Acrobat Reader, this article is available in hypertext, non-linear style where you can jump from a page to another, easily. However, if you are reading this article from a hardcopy printout, then you are reading this article in traditional, non-linear style, and you have to flip through the pages manually.

## 2. Voice-over narration and background music

Most of the times, voice-over narration (WAVE files) are played to explain and complement the screen shown. Background music (MIDI files) are used to create the impact of the demo, similar to watching a movie scene where sometimes, the music creates the ambience for excitement.

### 2.1 WAVE files

I have used WAVE files for both voice-narration and background music. As most kiosk display system are running on 'small' desktop speakers, it doesn't matter whether the WAVE files are originally encoded in 16-bit (high quality) or 8-bit (low/medium quality) format. For voice-over narration, it's a total waste of disk space if you encoded the WAVE files in 16-bit, stereo format as the sound output heard on the speakers is almost similar to those WAVE files encoded in 8-bit mono format.

However, as mentioned earlier, I have also used WAVE files for background music. In this case, the WAVE music files were either encoded to 8-bit (stereo) format, or in some cases, 16-bit stereo format as a stereo music output is able to create the surround effects (*left and right channel*) better than a mono-encoded music. Although WAVE files are essential larger in file size than MIDI files, there are more pros than cons, if disk space is not a constraint here. Here's a scenario:

Let's say a kiosk demo is to be deployed at 2 different locations, and the recommended system configuration is as follows:

1. *Pentium 133 or higher system*
2. *12x CD-ROM drive*
3. *Sound Blaster AWE32 or AWE64*
4. ..

Item 3 above states that the kiosk system is recommended to have a Creative Sound Blaster AWE32 or Sound Blaster AWE64 sound card installed in the system. This means that the sound (especially background MIDI music) used in

the kiosk demo were based on the assumption that either one of these 2 sound cards are used. What would happen if a Sound Blaster 16 or some other 3<sup>rd</sup>-party sound card is installed in the system, and/or the MIDI mapping was set to FM instead of GM? Well, the music heard will sound metallic, and will not create the musical impact it had originally meant to.

## 2.2 MIDI files

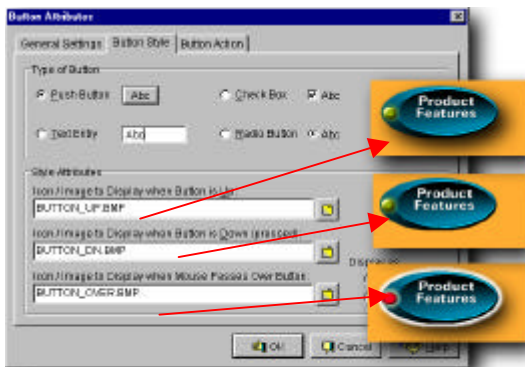
Use MIDI files when disk space is a constraint. Or when you are not really bothered whether the MIDI setting is set to FM or GM. That is, there will be no difference in the sound quality heard. Furthermore, playing back MIDI files usually uses less memory overheads than WAVE files. As a matter of fact, I have also used MIDI files in my kiosk demo on some pages.

## 3. Interactive, click-able 'push' buttons

Imagine you are now watching a kiosk demo with buttons for selection. You clicked at one of the buttons and a new screen appears immediately.

Next, imagine you are watching another kiosk demo with buttons for selection. You clicked at one of the buttons. There was a 'click' sound heard and the button is depressed like a 3-dimensional virtual button). Then a new screen appears.

Generally, the second kiosk demo will create a greater impact than the first demo. I would suggest you to take advantage of the 'Style Attributes' of the BUTTON ATTRIBUTES option where you can specify an image to display when the button is 'up', and another image when the button is 'down'. This will create a 'push' button effect. (See snapshot on the right). NBW v3.2 includes an additional 3<sup>rd</sup> option to allow you to specify an icon/image to display when the mouse passes over the button. This is useful especially when you need to 'highlight' the button to emphasize it. Another good example is the 'Standard Buttons Toolbar' in Microsoft Internet Explorer v4.0. If you move the mouse on top of these 'gray' buttons, a colored version will be displayed.



### 4. Self-running slide-show after a certain period of inactivity

The IDLE EVENT of BOOK SETUP allows you to specify the time (in minutes) of inactivity to activate an Action script. In my case, I have directed my demo to go to a SLIDESHOW page. (See Section 2 for more information.)

### 5. Individual time-event to 'return' to specific page upon completion of a task.

What this means is that you can specify your own 'idle time event' in any one page, or in any of the buttons in a page. Yes, who says it cannot be done in NBW? See Section 2 for more information.

### 6. Displaying of video clips within pre-define window area and/or option to maximize it to a full screen

In my cases, I have used the "MCICommand" Action Event command to play back DVD video clips saved in VOB format. (See Section 3 for more information)

### 7. Animations (automatic or manual)

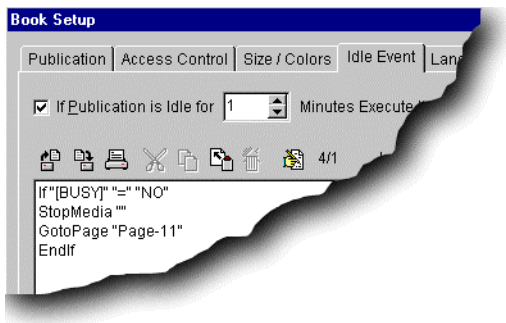
Taking advantage of the animation in NeoToons, you can add a NeoToon (\*.CAR) file to create some kind of animation. (More details on this will be discussed on future updated version of this article; or as a separate article)



## Section 2: Idle Event

The IDLE EVENT (in BOOK SETUP) currently allows you to define the time taken before the Action event takes place when there is no keyboard activity or mouse movement for a certain period of time (*in minutes*).

In my kiosk publication, the IDLE EVENT is set at 1-MINUTE. (See Figure 3)



[Figure 3]

The "Actions event" basically issued an instruction to stop playing all media (video, sound, and animation), and then proceed to 'PAGE 11' where the self-running slideshow starts. Initially, my plan was to develop 2 separate components in my kiosk demo (*i.e.*, 2 separate NBW publications to execute – one for the main interactive version; and the other, for the slideshow version).

The following factors were considered:

- When the DEMO.EXE runs, and then execute the SELFRUN.EXE, is the DEMO.EXE still in memory or only a basic 'shell' code is left so that when SELFRUN.EXE ends, it exits back to DEMO.EXE? An analogy is using the MS-DOS shell in Windows 95; running a DOS program, and then exits back to Windows environment. If this is true, the available memory will be reduced. And in such cases, by how much? If the DEMO.EXE is about 9MB and the SELFRUN.EXE is about 4MB, how much memory or RAM is required?
- Increase in disk space. In my case, this is not an issue as the final kiosk demo is 'running' from a CD-ROM disc.
- Unnecessary delay in loading time. When DEMO.EXE executes the SELFRUN.EXE, there will be unnecessary delay while loading the program into the system memory, especially when SELFRUN.EXE is more than 5 MB in file size.
- What is the maximum number of user-definable variables allowed in NBW?

As the slideshow screens in all the pages of my kiosk demo were basically identical to the interactive screens, I have decided to stick to one single publication. That is, PAGE 1 to PAGE 10 are for interactive version; and PAGE 11 to PAGE 20 are for the self-running version – although the screens in PAGE 1 to PAGE 10 are identical to PAGE 11 to PAGE 20 respectively.

Upon 1-minute inactivity (let's say the current displayed screen may be at PAGE 5), the demo will automatically performed the actions stated in the IDLE TIME dialog. This means that PAGE 11 will be displayed at a certain time, and then continue to PAGE 12, and so on, until it reaches PAGE 20. Should there still be no activity on the keyboard/mouse, the next PAGE to be displayed will be PAGE 11, and the self-running portion will run in a loop. However, should there be some keyboard activity, the slideshow will stop and return to the MAIN MENU (*i.e.*, PAGE 1). By using this method, I do not have to worry about the 4 factors mentioned on the previous page. Oh, check the FAQ Section for NeoSoft's reply.

### Creating individual IDLE time event

Having an idle event set only in MINUTES may be useful for most publications, but sometimes, it may be too long a period specified for inactivity for kiosk-like publications. The ideal solution is to have the IDLE EVENT made available in SECONDS instead of MINUTES. Of course, as you know, the IDLE EVENT is only available in MINUTES.

What if you have some pages displaying video (AVI, MOV, MPG, VOB) images? Well, you can simply specified in the IDLE EVENT the time (say, in MINUTES) required based on the longest playing video file. For example, the longest video file (SAMPLE.AVI) is about 2 minutes and 55 seconds, you can simply specified the IDLE EVENT to 3-minutes. This means that upon the 3-minutes time frame, the IDLE EVENT will invoke the script for the next action to take place.

Well, again it is not practical to specify the IDLE EVENT simply based on the longest possible time. What if you have a static page; or a page that displays a short video of say, 30 seconds? These pages will not move for more than 2 minutes, if the user walks away.

The next ideal solution is to have a user-definable IDLE EVENT that allows you to determine individual timing for individual pages, and yet at the same time, specifying a standard time length in the 'actual' IDLE EVENT (BOOK SETUP).

*Confuse?*



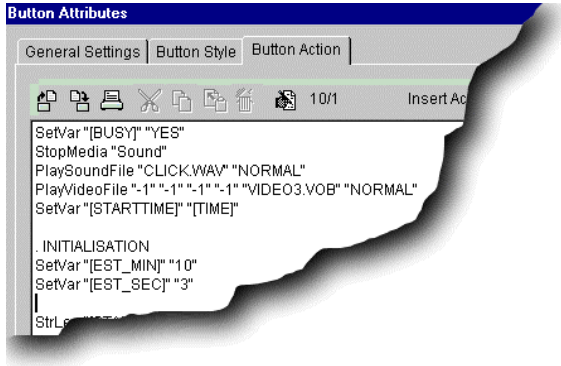
[Figure 4]

In the above screen snapshot, there are 3 button-options available to display 3 different video clips in the 'SAMPLE VIDEO CLIPS' page. Let's say the time-length for the 3 video clips are as follows:

VIDEO1.AVI	- 1 minute 6 seconds
VIDEO2.MOV	- 2 minutes 58 seconds
VIDEO3.VOB	- 10 minutes 3 seconds

This means that when a user clicks at VIDEO1 button, it will display VIDEO1.AVI in the window (on the right). And assuming the IDLE EVENT is set at 1 Minute. This means that when VIDEO1.AVI stops, it will do nothing until 54 seconds later, when the IDLE EVENT activates. And what if a user clicks at VIDEO3 button, and stand and watch the VIDEO3.VOB video clip played back. What do you think will happen? This video clip will stop when it's at the 1-minute time frame, and starts activating the IDLE EVENT Action script. What do you think will happen? Chances are, the user will feel agitated and walks away from the kiosk system.

In my case, I have specified the IDLE TIME specified to be the lowest possible value, i.e. 1-minute. This means that for every 1 minute, the IDLE EVENT Action script will be activated should there be no activity on the keyboard or mouse. To avoid the consequences indicated in the example mentioned in the previous paragraph, you will need to look into the 4 elements carefully, and specified each element with a different time event. That is to say, we need to specify the time event individually for VIDEO1, VIDEO2 and VIDEO3. The 4<sup>th</sup> element (IDLE EVENT) is already taken care of in this case through the BOOK SETUP menu option (See Figure 3).



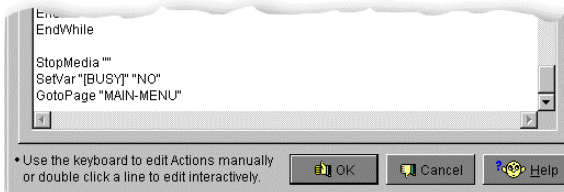
[Figure 5]

If you observed from Figure 4 above, there is a [BUSY] variable being used. This is a GLOBAL variable where the value is indicated as either 'YES' or 'NO'. By issuing the instruction to check whether `If "[BUSY]" "=" "NO"` in the IDLE EVENT script (Figure 3), I can then determine whether a particular page or button action is currently busy performing a task or not. If it is not, the Action script indicated will be executed.

To tell the program that a particular action/task is busy, simply add the following statement:

```
SetVar "[BUSY]" "YES"
```

As in the case for the VIDEO3 button, you can stop the IDLE EVENT Action script from being activated, even after 1-minute keyboard inactivity as the [BUSY] is set to "YES" (Figure 5). The IDLE EVENT will then ignore the Action script, and re-initialize the timing, and checks whether the [BUSY] is "NO". And if so, it will then execute the Action script.

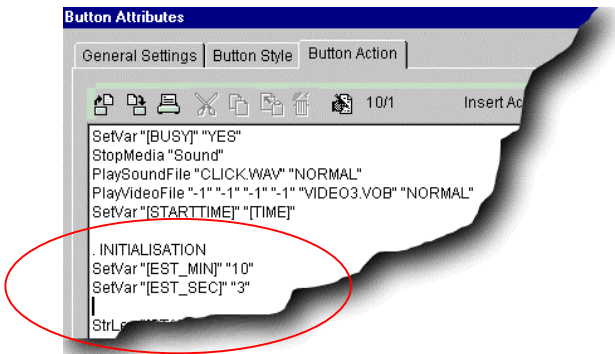


[Figure 6]

## DEVELOPING A MULTIMEDIA KIOSK WITH NEOBOOK

What this means is that when VIDEO3.VOB is played, the video will not stopped after 1 minute. Instead, it will continue displaying until the 10 minutes and 3 seconds time frame is up. Then it will continue with the next instruction, i.e., set [BUSY] = "NO", and then [GotoPage "MAIN-MENU"](#) (See Figure 6).

What this means is that I can now finally specify individual time event for VIDEO1 and VIDEO2 buttons as well, in addition to the default 1 minute for some other static screens. This makes the kiosk demo is 'intuitive' enough to return to another page after completion of task; or after a certain period of keyboard/mouse inactivity. Of course, at any time while the video is playing, other buttons should be available to STOP the playback and proceed to the next course of action.



NOTE: There is an INITIALISATION routine that checks whether the time (i.e., 10 minutes and 3 seconds) is up while the video clip (VIDEO3.VOB) is being played.

### For NeoBook v3.1c only

NBW 3.1c only provides a [TIME] variable. Strictly in computer terms, you cannot directly use the current time, e.g., "9:09:05 PM" and subtract it from the starting time, e.g., "8:59:20"; and expect to come up with an answer for the total time lapse to be "1:10:45".

Assuming I need to create a procedure to determine the total time lapse. In theory, this means [CURRENT\_TIME – START TIME = TOTAL TIME LAPSE]. Because there is only a [TIME] variable, I have to resort to using [SUBSTR] variable to determine the total length of the resulting time and the particular position each for the HOUR, MINUTES and SECONDS. I'll also have to consider that if the SECONDS (e.g., "09"), I have to truncate the trailing '0' as in '09' and the result for SECONDS is a single '9' integer. This is because the computer is not smart enough to understand the mathematical equation [9:09:05 – 8:59:20 = 10:45 mins]. Instead, it will result in an error, and/or will not compute correctly. In a typical worst scenario, the program will be running in an 'Endless-loop' and the only method to stop it is to reset the program.

*This means my code for checking the MINUTES and SECONDS are like this:*

Assuming the start time is initialized somewhere when the button is clicked. The [BUSY] variable (in Line 0) is used here as a global variable, and is set to "YES". This means when the following codes are executed, especially Line 3 where a video clip is currently being played, the IDLE EVENT (set to 1-minute) will not be invoked as the Action scripts (in the IDLE EVENT) will only be executed if the [BUSY] variable is set to "NO".

```
Line 0      SetVar "[BUSY]" "YES"  
Line 1      SetVar "[START_TIME]" "[TIME]"  
Line 2      PlayVideoFile "-1" "-1" "-1" "-1" "VIDEO2.MOV" "NORMAL"
```

As the time-length for VIDEO2.MOV is 2 minutes 58 seconds, we'll need to create 2 user-variables to store the estimated minutes and seconds.

```
Line 3      SetVar "[EST_MIN]" "2"  
Line 4      SetVar "[EST_SEC]" "58"
```

Now we need to calculate length [SIZE] in the result of the starting time, and then check the individual position for the minutes and seconds. For example, '5' is one character length and '15' is 2 characters length.

```
Line 5      StrLen "[START_TIME]" "[SIZE]"
```

If the total length size for starting time is 10 characters (e.g., 1:25:30 PM), the following instructions will be executed to check for the first starting point where the minutes start. In this case, it's at the 3<sup>rd</sup> position. Here, the [TVAR] variable is used as a temporary user-variable.

## DEVELOPING A MULTIMEDIA KIOSK WITH NEOBOOK

```
Line 6      If "[SIZE]" "=" "10"  
Line 7          SubStr "[START_TIME]" "3" "2" "[START_MIN]"  
Line 8          SubStr "[START_MIN]" "1" "1" "[TVAR]"
```

If the 3<sup>rd</sup> position of the minutes is '0', we will ignore this position and set the starting point to be at 4<sup>th</sup> position instead so that we can obtain a valid integer, e.g., "9" and not "09".

```
Line 9      If "[TVAR]" "=" "0"  
Line 10          SubStr "[START_TIME]" "4" "1" "[START_MIN]"  
Line 11      EndIf
```

Next, we will perform the same algorithm to check the position of the seconds. In this case, the starting point for the seconds is at the 6<sup>th</sup> position:

```
Line 12          SubStr "[START_TIME]" "6" "2" "[START_SEC]"  
Line 13          SubStr "[START_SEC]" "1" "1" "[TVAR]"
```

If the 6<sup>th</sup> position of the seconds is '0', we will ignore this position and set the starting point to be at 7<sup>th</sup> position instead so that we can obtain a valid integer, e.g., "5" and not "05".

```
Line 14      If "[TVAR]" "=" "0"  
Line 15          SubStr "[START_TIME]" "7" "1" "[START_SEC]"  
Line 16      EndIf
```

```
Line 17      Else
```

Similarly, if the length is 11 (e.g., 11:25:40 PM), the starting point for the minutes is at the 4<sup>th</sup> position:

```
Line 18          SubStr "[START_TIME]" "4" "2" "[START_MIN]"  
Line 19          SubStr "[START_MIN]" "1" "1" "[TVAR]"
```

If the 4<sup>th</sup> position of the minutes is '0', we will ignore this position and set the starting point to be at 5<sup>th</sup> position instead so that we can obtain a valid integer, e.g., "9" and not "09".

```
Line 20      If "[TVAR]" "=" "0"  
Line 21          SubStr "[START_TIME]" "5" "1" "[START_MIN]"  
Line 22      EndIf
```

Next, we will perform the same algorithm to check the position of the seconds. In this case, the starting point for the seconds is at the 7<sup>th</sup> position:

```
Line 23          SubStr "[START_TIME]" "7" "2" "[START_SEC]"  
Line 24          SubStr "[START_SEC]" "1" "1" "[TVAR]"
```

If the 7<sup>th</sup> position of the seconds is '0', we will ignore this position and set the starting point to be at 8<sup>th</sup> position instead so that we can obtain a valid integer, e.g., "5" and not "05".

## DEVELOPING A MULTIMEDIA KIOSK WITH NEOBOOK

```
Line 25         If "[TVAR]" "=" "0"  
Line 26             SubStr "[START_TIME]" "8" "1" "[START_SEC]"  
Line 27         EndIf  
Line 28     EndIf
```

To obtain the [MIN\_REQ], the following mathematical calculations are performed, as 60 minutes make 1 hour. For example, using the below example, the [MIN\_REQ] is "60-2", i.e., 58.

```
Line 29     Math "60 - [EST_MIN]" "0" "[MIN_REQ]"
```

Here's where the comparison starts. If the [START\_MIN] is less than the [MIN\_REQ], add the estimated [EST\_MIN] to the [START\_MIN]. Using our example, this means that if the [START\_MIN] is less than 58, i.e., 57 and below, we can safely add the estimated [EST\_MIN] (i.e., 2 minutes) to the [TOTALMIN], as the total value will be less than 60.

```
Line 30     If "[START_MIN]" "<" "[MIN_REQ]"  
Line 31         Math "[START_MIN]+[EST_MIN]" "0" "[TOTALMIN]"  
Line 32     EndIf
```

If the [START\_MIN] is 58, and the [MIN\_REQ] is also 58, the [TOTALMIN] value in this example will be '0'.

```
Line 33     If "[START_MIN]" "=" "[MIN_REQ]"  
Line 34         SetVar "[TOTALMIN]" "0"  
Line 35     EndIf
```

If the [START\_MIN] is more than '58' (i.e., '59'), and the [EST\_MIN] is '2', we know that the result is '61', which is more than '60'. In this case, we need to use the result and subtract it by '60', so that the end-result will be stored in [TOTALMIN]. In this example, the [TOTALMIN] is '1'

```
Line 36     If "[START_MIN]" ">" "[MIN_REQ]"  
Line 37         Math "([START_MIN]+[EST_MIN])-60" "0" "[TOTALMIN]"  
Line 38     EndIf
```

Now, we will calculate the [TOTALSEC] required by adding the [START\_SEC] with the [EST-SEC]:

```
Line 39     Math "[START_SEC]+[EST_SEC]" "0" "[TOTALSEC]"
```

If the [TOTALSEC] is more than '60', the instructions in Line 68 – Line 69 will be executed. For example, if the [TOTALSEC] is '75', by deleting it with 60', the correct [TOTALSEC] is now '15'. And because it's more than 60, we need to increment the [TOTALMIN] value by '1' as well.

```
Line 40     If "[TOTALSEC]" ">" "60"  
Line 41         Math "[TOTALSEC] - 60" "0" "[TOTALSEC]"  
Line 42         Math "[TOTALMIN] + 1" "0" "[TOTALMIN]"  
Line 43     EndIf
```

## DEVELOPING A MULTIMEDIA KIOSK WITH NEOBOOK

```
Line 44     If "[TOTALSEC]" "=" "60"  
Line 45         SetVar "[TOTALSEC]" "0"  
Line 46     EndIf
```

Now that we have calculated and obtained the valid integer for [START\_MIN] and [START\_SEC] of the [START\_TIME], we have to use the same concept to calculate the [CUR\_MIN] and [CUR\_SEC] of the [CUR\_TIME]. Before we continue, we need to set a looping variable [LOOPS] so that the current time is continuously checked, calculated, and compare with the results of [START\_TIME]. And once the values in [CUR\_MIN] and [CUR\_SEC] is 'equal' to the values in [TOTALMIN] and [TOTALSEC], the loop will end. As we are using the [SIZE] and [TVAR] as user-variables again, we need to re-initialize it.

```
Line 47     SetVar "[LOOPS]" "1"  
Line 48     SetVar "[SIZE]" ""  
Line 49     SetVar "[TVAR]" ""
```

As LOOPS has been initialized to '1', the below statement will continue looping until the 'LOOPS' variable becomes '0'

```
Line 50     While "[LOOPS]" "<>" "0"
```

Check current time

```
Line 51     SetVar "[CUR_TIME]" "[TIME]"  
Line 52     StrLen "[CUR_TIME]" "[SIZE]"
```

The algorithm used here is similar to the one provided to check the [START\_TIME]:

```
Line 53     If "[SIZE]" "=" "10"  
Line 54         SubStr "[CUR_TIME]" "3" "2" "[CUR_MIN]"  
Line 55         SubStr "[CUR_MIN]" "1" "1" "[TVAR]"  
  
Line 56         If "[TVAR]" "=" "0"  
Line 57             SubStr "[CUR_TIME]" "4" "1" "[CUR_MIN]"  
Line 58         EndIf  
  
Line 59         SubStr "[CUR_TIME]" "6" "2" "[CUR_SEC]"  
Line 60         SubStr "[CUR_SEC]" "1" "1" "[TVAR]"  
  
Line 61         If "[TVAR]" "=" "0"  
Line 62             SubStr "[CUR_TIME]" "7" "1" "[CUR_SEC]"  
Line 63         EndIf  
  
Line 64     Else  
  
Line 65         SubStr "[CUR_TIME]" "4" "2" "[CUR_MIN]"  
Line 66         SubStr "[CUR_MIN]" "1" "1" "[TVAR]"  
  
Line 67         If "[TVAR]" "=" "0"  
Line 68             SubStr "[CUR_TIME]" "5" "1" "[CUR_MIN]"  
Line 69         EndIf  
Line 70         SubStr "[CUR_TIME]" "7" "2" "[CUR_SEC]"
```

## DEVELOPING A MULTIMEDIA KIOSK WITH NEOBOOK

```
Line 71      SubStr "[CUR_SEC]" "1" "1" "[TVAR]"

Line 72      If "[TVAR]" "=" "0"
Line 73      SubStr "[CUR_TIME]" "8" "1" "[CUR_SEC]"
Line 74      EndIf
Line 75      EndIf
```

Here's where the actual comparison algorithm is use to determined whether the total time taken is identical to the current time:

```
Line 76      If "[CUR_MIN]" "=" "[TOTALMIN]"
Line 77      If "[CUR_SEC]" "=" "[TOTALSEC]"
```

If it matches, we'll need to set the [LOOPS] variable to '0' so that it can issue a 'control break' to stop and 'end' the WHILE-ENDWHILE loop procedure, and continue from Line 77.

```
Line 78      SetVar "[LOOPS]" "0"
Line 79      EndIf
```

If the current time is still not identical to the total time, the process will be repeated from Line 32 until the current time is identical to the total time.

```
Line 80      EndIf
Line 81      EndWhile
```

Here's where you would continue your next course of action. In this example, we will stop playing and close the VIDEO2.AVI file; and return to the 'SUB-MENU' page.

```
Line 82      StopMedia ""
Line 83      SetVar "[BUSY]" "NO"
Line 84      GotoPage "SUB-MENU"
```

*NOTE: Since NBW v3.2 is now released, the coding and implementation used for checking the time, etc., is now considered 'REDUNDANT'. I have included them here as I think they can be useful for some other task where you need to check the length, position and perform some calculations based on a user variable. A 'cleaner' code listing with fewer comments is available in Appendix I.*

## DEVELOPING A MULTIMEDIA KIOSK WITH NEOBOOK

### For NeoBook v3.2

It's easier to code with the NBW v3.2 as the following internal variables - [Hour] [Minute] and [Second] are now supported. Furthermore, the trailing '0' (zero) is automatically truncated and the result is a valid integer. Let's say the current time is '3:09:08 PM', the results will be as follows:

```
[Hour]    = 3
[Minute]  = 9
[Second]  = 8
```

In this case, when VIDEO2.MOV video clip reaches the last frame and stop, it will be about '3:12:06'.

*Note: Although we all know that it takes 60 seconds to make 1 minute; and when the SECONDS digit is 59, the next digit is 0 (zero). So how do we instruct the computer (or NBW) to do a mathematical calculation and also ensure that the SECOND digit resets to 0 after 59?*

*This means my code for checking the MINUTES and SECONDS are like this:*

Assuming the start time is initialized somewhere when the button is clicked. If you observed, I've do away with the [TIME] variable and make full use of the [MINUTE] and [SECOND] instead. The [BUSY] variable (in Line 0) is used here as a global variable, and is set to "YES". This means when the following codes are executed, especially Line 3 where a video clip is currently being played. The IDLE EVENT (set to 1-minute) will not be invoke as the Action scripts (in the IDLE EVENT) will only be executed if the [BUSY] variable is set to "NO".

```
Line 0    SetVar "[BUSY]" "YES"
Line 1    SetVar "[START_MIN]" "[MINUTE]"
Line 2    SetVar "[START_SEC]" "[SECOND]"
Line 3    PlayVideoFile "-1" "-1" "-1" "-1" "VIDEO2.MOV" "NORMAL"
```

As the time-length for VIDEO2.MOV is 2 minutes 58 seconds, we'll need to create 2 user-variables to store the estimated minutes and seconds.

```
Line 4    SetVar "[EST_MIN]" "2"
Line 5    SetVar "[EST_SEC]" "58"
```

As 60 minutes make 1 hour, it is important to obtain the correct MINUTES required (MIN\_REQ) for comparing, by using the following mathematical calculations. For example, using the below example, the [MIN\_REQ] is "60-2", i.e., 58.

```
Line 6    Math "60 - [EST_MIN]" "0" "[MIN_REQ]"
```

Here's where the comparison starts. If the current MINUTE integer is less than the compared [MIN\_REQ] integer, add the estimated MINUTE to the current [MINUTE]. Using our example, this means that if the MINUTE is less than 58,

## DEVELOPING A MULTIMEDIA KIOSK WITH NEOBOOK

i.e., 57 and below, we can safely add the estimated minute (i.e., 2 minutes) to the current MINUTE, as the total will not be more than 59.

```
Line 7   If "[START_MIN]" "<" "[MIN_REQ]"
Line 8       Math "[START_MIN]+[EST_MIN]" "0" "[TOTALMIN]"
Line 9       EndIf
```

If the current MINUTE is 58, and the compared [MIN\_REQ] is also 58 in this example, do:

```
Line 10  If "[START_MIN]" "=" "[MIN_REQ]"
Line 11      SetVar "[TOTALMIN]" "0"
Line 12  EndIf
```

If the current minute is more than 58, say 59, we will need to add the [MINUTE] (59) and the [EST\_MIN] (2). The result will be 61. Then subtract this result with 60 so that the total minute required is  $(61-60=1)$  is 1 minute.

```
Line 13  If "[START_MIN]" ">" "[MIN_REQ]"
Line 14      Math "([START_MIN]+[EST_MIN])-60" "0" "[TOTALMIN]"
Line 15  EndIf
```

Now, we will perform the same calculation for SECONDS.

```
Line 16  Math "[START_SEC]+[EST_SEC]" "0" "[TOTALSEC]"
```

In our case, the estimated SECONDS is 58, and it's less than 60, we can safely ignore the procedure Line 69 to 70, and continue from Line 71.

```
Line 17  If "[TOTALSEC]" ">" "60"
Line 18      Math "[TOTALSEC] - 60" "0" "[TOTALSEC]"
Line 19      Math "[TOTALMIN] + 1" "0" "[TOTALMIN]"
Line 20  EndIf
```

```
Line 21  If "[TOTALSEC]" "=" "60"
Line 22      SetVar "[TOTALSEC]" "0"
Line 23      Math "[TOTALMIN] + 1" "0" "[TOTALMIN]"
Line 24  EndIf
```

Before we continue, we need to set a looping variable [LOOP] so that the current time is continuously checked, calculated, and compare with the results of START time. And once the estimated time is equal to CURRENT, the loop will end.

```
Line 25  SetVar "[LOOPS]" "1"
```

As LOOPS has been initialized to '1', the statement will continue looping until the 'LOOPS' variable becomes '0'

```
Line 26  While "[LOOPS]" "<>" "0"
```

Check current time

```
Line 27  SetVar "[CUR_MIN]" "[MINUTE]"
```

## DEVELOPING A MULTIMEDIA KIOSK WITH NEOBOOK

```
Line 28 SetVar "[CUR_SEC]" "[SECOND]"
```

Here's where the actual comparison procedure is determined whether the current time taken is identical to the estimated time. If the total minutes and total seconds are identical, do:

```
Line 29 If "[CUR_MIN]" "=" "[TOTAL_MIN]"  
Line 30 If "[CUR_SEC]" "=" "[TOTALSEC]"
```

We'll need to reset the LOOPS variable to '0' will initiates a 'control break' to stop the WHILE loop. And if the [LOOPS] becomes '0', this means the WHILE-DO condition is true. In this case, the program will jump to Line 80.

```
Line 31 SetVar "[LOOPS]" "0"  
Line 32 EndIf
```

If the total minutes and total seconds are still not identical to the estimated time, we will have to repeat the LOOP process and check until the current time matches the total time.

```
Line 33 EndIf  
Line 34 EndWhile
```

Here's where you would continue your action. In this example, we will stop and close the VIDEO2.AVI file, and return to the previous SUB-MENU page.

```
Line 35 StopMedia ""  
Line 36 SetVar "[BUSY]" "NO"  
Line 37 GotoPage "SUB-MENU"
```

*Note: Using the same algorithm, the number of coding lines required for NBW v3.2 is much lesser than NBW v3.1c, resulting in a slightly smaller compiled NBW file. A 'cleaner' code listing with fewer comments is available in Appendix 2.*

## **Section 3 – Using Action Event Commands**

### **MCI Command**

Basically, the “MCICommand” command is used to send an instruction to the Windows Media Control Interface (MCI) to perform specific task. What this means is that technically and theoretically, you can access any hardware devices (e.g., CD-ROM drive, Laser-Disc, DVD Players) as long as these devices support the MCI by providing the necessary MCI device drivers.

As NBW currently does not provide direct support to PC-DVD players, I have to resort to using the “MCICommand” to perform the following task:

Opening a video file (\*.VOB)

Define a window area (coordinates) for the video to be displayed

Hide the window handle

Play the video file

Stop the video file

Close the video file



Let's say we have a page as shown in the above snapshot. As you can see, there are 3 video button options to allow a user to click. If a user clicks at any of the button, the appropriate video clip will be displayed within the window area shown on the right portion.

Using VIDEO 1 button as an example, my code will be like this:

```
SetVar "[BUSY]" "YES"
```

The above [BUSY] variable is used to tell the IDLE EVENT that the VIDEO1 button is busy performing a task, i.e., about to play back a video clip. And this will not activate the IDLE EVENT after the 1-minute time frame.

```
MCICommand "open VIDEO1.VOB alias VIDEOfILE"  
MCICommand "put VIDEOfILE window at 238 79 519 405"  
MCICommand "window VIDEOfILE state hide"  
MCICommand "play VIDEOfILE"
```

## DEVELOPING A MULTIMEDIA KIOSK WITH NEOBOOK

The 'alias' used here basically allows me to refer the "VIDEO1.VOB" file as "VIDEOFILE" instead. This means that whenever I want to access the 'VIDEO1.VOB' file, I'll simply use the "VIDEOFILE" alias name instead.

If I did not specify an 'alias' name, and if the video file is stored in the different directory, this means my action script will be like this:

```
MCICommand "open C:\DEMO\VIDEO\CLIPS\VIDEO1.VOB"  
MCICommand "put C:\DEMO\VIDEO\CLIPS\VIDEO1.VOB window at 0 0 200 500"  
MCICommand "window C:\DEMO\VIDEO\CLIPS\VIDEO1.VOB state hide"  
MCICommand "play C:\DEMO\VIDEO\CLIPS\VIDEO1.VOB"
```

Apart from lengthy statement, and chances of 'typo' errors, it uses more bytes as well, resulting in unnecessary disk space. There is also an advantage of using an alias name, i.e. VIDEOFILE in this case, as I can use the alias as a GLOBAL variable.

### . INITIALIZE

```
SetVar "[EST_MIN]" "2"  
SetVar "[EST_SEC]" "59"
```

As you see, I have consistently used the 'INITIALIZE' routine as well, which I have mentioned earlier. Refer to "Section 2: IDLE EVENT" for more information. Likewise, the statement below will stop and close the VIDEOFILE after the current time is identical to the estimated time. The [BUSY] variable is reset to "NO" to allow the IDLE EVENT to take place.

```
MCICommand "stop VIDEOFILE"  
MCICommand "close VIDEOFILE"  
SetVar "[BUSY]" "NO"
```

The above routine is ideal if there is no user intervention via keyboard or mouse control while the video is being played back. If the user is there, and clicks at the 'STOP' button or even say, VIDEO2 button, the following statement must be included to stop and close the VIDEO1.VOB file:

```
MCICommand "stop VIDEOFILE"  
MCICommand "close VIDEOFILE"  
SetVar "[BUSY]" "NO"
```

Otherwise, if the VIDEO1.VOB file is not stopped and closed, there may be problems like VIDEO2 overlapping VIDEO1 resulting in distorted display; error or system hang due to improper opening/closing statements.

The [BUSY] variable has to be reset to "NO". We can assume the user is still standing at the kiosk demo, and this user may just walk away after say, clicking at the STOP button. If we don't reset the [BUSY] variable to 'NO', the IDLE EVENT Action cannot take place, and the screen will remain at that page until a button is clicked, or after the next system reboot.

## DEVELOPING A MULTIMEDIA KIOSK WITH NEOBOOK

Below is a brief description of some of the MCI Commands for your reference. As different MCI device may use different implementation in their drivers (*i.e., the order of sequence for instruction/command may be different from one MCI device to another*), the below information is purely for information, and may not work with the MCI device that you are accessing. I would suggest you check the Microsoft SDK for more detailed MCI commands. Anyway, the syntax to use in NBW is as follows:

MCICommand "<MCI Command> <Filename/Alias> <Parameters>

Using the same coding mentioned earlier, to open a file via MCI, use:

MCICommand "open VIDEO1.VOB alias VIDEOfILE"

Now, I need to put this file within a pre-defined window area:

MCICommand "put VIDEOfILE window at 238 79 519 405"

After that, I need to hide the window 'handle', so that the window border are hidden:

MCICommand "window VIDEOfILE state hide"

To play the file and display the video within the window:

MCICommand "play VIDEOfILE"

MCI COMMAND	Parameters	Description
<b>Configure</b>		
<b>Info</b>	File Product Window text	
<b>Open</b>	<filename> Alias	
<b>Pause</b>		
<b>Play</b>	Notify Wait From To	
<b>Put</b>	Destination at <RECT> Window at <RECT>	
<b>Resume</b>		

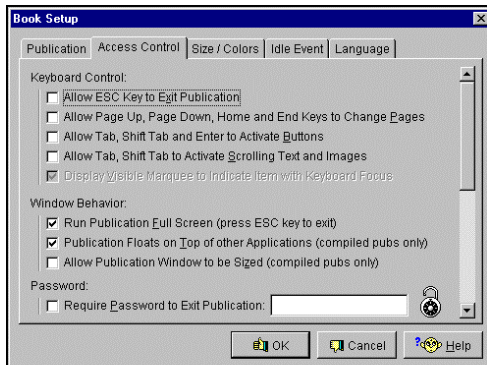
## DEVELPING A MULTIMEDIA KIOSK WITH NEOBOOK

<b>Seek</b>	To	
	To start	
<b>Set</b>	Time format ms	
	Time format frames	
	Time format bytes	
<b>Setaudio</b>	On	
	Off	
	Volume to <0 – 100>	
<b>Setvideo</b>	On	
	Off	
	Video key color to	
<b>Status</b>	Length	
	Mode	
	Position	
	Time format	
	Handle	
	Video key color	
<b>Stop</b>		
<b>Where</b>	Window	
	Window max	
	Destination	
	Source	
<b>Window</b>	Handle	
	State show	
	State maximized	
	State minimized	
	State hide	
	Show normal	
	Minimize	
	Restore	

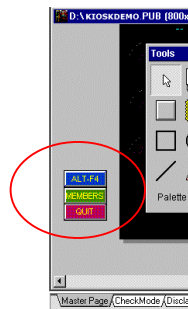
## TIPS & Tricks

### 1. Setting up Keyboard controls

The ACCESS CONTROL of BOOK SETUP (see below) allows you to configure the way you want your publication to work.



As my publication is a kiosk demo, I have enabled only the first 2 options of the “Window Behavior” option. Be careful if you enabled the “Allow ESC key to Exit Publication” and forgot to enable the “Password” check box, and supplied the password. While doing my demo, I’ve observed that although the above options work fine, there is a problem: You can still press ALT-F4 to exit the program. What I did was to create a “ALT-F4” button in the MASTER PAGE (see snapshot on right) and the “Short Cut Key” is specified as “ALT + F4”, and the Button Action as ‘Delay “5”’. What this means is that as the button is placed outside the publication, it will not be displayed on the screen when run, and users are not aware of the existence of such button. When a ‘naughty’ user press ALT-F4, it basically executes the ‘DELAY “5”’ instruction, and does nothing. From the user’s point of view, nothing will appear on the screen, and it seems like the demo will ‘trapped’ the ALT-F4 keystroke as well. Oh, one point to note – there may be pages where you may disabled the “Show Master Items” from the BOOK option. In such cases, you have to copy the ‘ALT-F4’ button to these pages as well. Otherwise, the publication will exit if user happen to press ‘ALT-F4’ from this particular page!



If you are observant enough, I have also added a ‘QUIT’ button. This button allows only authorized people (those who know the “Short Cut Key”) to exit the demo. Preferably, use 3 keystrokes, e.g., [CTRL + SHIFT + PGDN] to prevent

the demo from exiting unnecessary, or without proper authorization. With the combinations of 3 keystrokes... it's not easy to know which keys are used to exit the program.

## 2. Running publication at specified screen mode

If you want to 'force' the system to be configured at the same display resolution and color depth (e.g., 800 X 600 with 16-bit), the "SystemInfo" command is used. From the User Guide or online help of NBW (v3.2), the "SystemInfo" command syntax is indicated as follows:

**Syntax:** SystemInfo "information type" "[variable name]"

Example:

```
SystemInfo "ScreenColors" "[ScreenMode]"
SystemInfo "PubColors" "[PubMode]"
If "[PubMode]" ">" "[ScreenMode]"
    MsgBox "Notice!" "This program requires [PubMode] colors in order to
display|properly. Since Windows is currently running in [ScreenMode]
color|mode, then you will not be viewing this program under|optimal conditions."
EndIf
```

In NeoBook 3.2, you could simplify this script by referencing the ScreenColors and PubColors variables directly:

```
If "[PubColors]" ">" "[ScreenColors]"
    MsgBox "Notice!" "This program requires [PubColors] colors in order to
display|properly. Since Windows is currently running in [ScreenColors]
color|mode, then you will not be viewing this program under|optimal conditions."
EndIf
```

**NOTE:** The [ScreenColors] only checks for color depth. You'll still have to use [ScreenHeight] to check for the screen resolution.

Unfortunately, this command cannot be used from the "Master Page" of the publication. As such, the command will usually be placed on Page 1 instead, so that if the [ScreenHeight] and [ScreenColors] does not match those of the [PubHeight] and [PubColors], the messages specified in the "MsgBox" is displayed.

If you used the 'hidden buttons' described on previous page, and assuming the system is configured at 1024 x 768 (16-bit), the error message dialog will be displayed, together with the 'hidden buttons'. A workaround is to set the "Fill Pattern" of these buttons to "Hollow" and the "Line Width" as "None" so that these buttons appear to be 'transparent'. Even if there is an error, these buttons are 'transparent' and no one knows that there are actually some 'hidden' buttons on the screen.

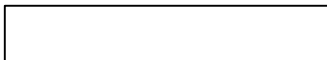
### **3. Including Fonts in your compiled publication**

We cannot take for granted that on everyone's system, there is a typeface called 'ARIAL' or 'VERDANA'. Even if these typefaces are already installed in the system, we cannot assume that the characteristics of these fonts (bold, italics, kerning, point sizes) in the same typeface are identical to the ones installed in your system where you compiled your NBW publication. Sometimes, an ARIAL typeface on one system may differ from an ARIAL typeface on another system. It happened to me before. Anyway, if you're not using any copyrighted or proprietary fonts/typeface in your publication, I would recommend that you include these fonts when compiling your publication using the 4<sup>th</sup> option from the COMPILE BOOK-FONTS option. In any case, do remember the following precautionary statement displayed in the option window:

**This option allows you to decide how fonts used in this publication appear when the compiled program is run on other computers. For maximum compatibility, fonts may be embedded inside the compiled program. This insures that the fonts you used to create the publication are the ones that your viewers will see. Although embedded fonts will not be accessible to other applications, you must check with your font vendor to insure that embedding fonts does not violate your license agreement.**

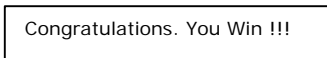
Let's say you have created a simple rectangular box in your publication to display some text with the variable as "[WINNER]" using Arial 9-points. Initially, the [WINNER] variable is 'empty', and no text is displayed in the box.

SetVar "[WINNER]" ""



Upon completion of a task, you changed the [WINNER]' variable and the text will be displayed in the box as:

SetVar "[WINNER]" "Congratulations. You Win !!!"



In most cases, you will expect the same results to be displayed on user's system. But on some systems, some words may be truncated (due to different characteristics of the particular font, although the name of the typeface in this case is also called 'ARIAL'); and the result may be displayed as:



## APPENDIX 1

### Specifying individual TIME event with NBW v3.1c

Statement lines 1 and 2 are important as it initializes the starting minutes and seconds. Change Line 3 accordingly to execute whatever commands. Remember to remove the LINE x words

```
SetVar "[BUSY]" "YES"
SetVar "[START_TIME]" "[TIME]"
PlayVideoFile "-1" "-1" "-1" "-1" "VIDEO2.MOV" "NORMAL"
```

Specify the minutes and seconds in following statement lines:

```
SetVar "[EST_MIN]" "2"
SetVar "[EST_SEC]" "58"
```

Here's where the algorithm starts. You can easily copy/paste it to your NBW publication. More explanation is available in Section 2.

```
StrLen "[START_TIME]" "[SIZE]"
If "[SIZE]" "=" "10"
    SubStr "[START_TIME]" "3" "2" "[START_MIN]"
    SubStr "[START_MIN]" "1" "1" "[TVAR]"

    If "[TVAR]" "=" "0"
        SubStr "[START_TIME]" "4" "1" "[START_MIN]"
    EndIf

    SubStr "[START_TIME]" "6" "2" "[START_SEC]"
    SubStr "[START_SEC]" "1" "1" "[TVAR]"

    If "[TVAR]" "=" "0"
        SubStr "[START_TIME]" "7" "1" "[START_SEC]"
    EndIf
Else
    SubStr "[START_TIME]" "4" "2" "[START_MIN]"
    SubStr "[START_MIN]" "1" "1" "[TVAR]"

    If "[TVAR]" "=" "0"
        SubStr "[START_TIME]" "5" "1" "[START_MIN]"
    EndIf

    SubStr "[START_TIME]" "7" "2" "[START_SEC]"
    SubStr "[START_SEC]" "1" "1" "[TVAR]"

    If "[TVAR]" "=" "0"
        SubStr "[START_TIME]" "8" "1" "[START_SEC]"
    EndIf
EndIf
```

```

Math "60 - [EST_MIN]" "0" "[MIN_REQ]"

If "[START_MIN]" "<" "[MIN_REQ]"
    Math "[START_MIN]+[EST_MIN]" "0" "[TOTALMIN]"
EndIf

If "[START_MIN]" "=" "[MIN_REQ]"
    SetVar "[TOTALMIN]" "0"
EndIf

If "[START_MIN]" ">" "[MIN_REQ]"
    Math "([START_MIN]+[EST_MIN])-60" "0" "[TOTALMIN]"
EndIf

Math "[START_SEC]+[EST_SEC]" "0" "[TOTALSEC]"

If "[TOTALSEC]" ">" "60"
    Math "[TOTALSEC] - 60" "0" "[TOTALSEC]"
    Math "[TOTALMIN] + 1" "0" "[TOTALMIN]"
EndIf

If "[TOTALSEC]" "=" "60"
    SetVar "[TOTALSEC]" "0"
EndIf

SetVar "[LOOPS]" "1"

SetVar "[SIZE]" ""
SetVar "[TVAR]" ""

While "[LOOPS]" "<>" "0"
    SetVar "[CUR_TIME]" "[TIME]"
    StrLen "[CUR_TIME]" "[SIZE]"

    If "[SIZE]" "=" "10"
        SubStr "[CUR_TIME]" "3" "2" "[CUR_MIN]"
        SubStr "[CUR_MIN]" "1" "1" "[TVAR]"

        If "[TVAR]" "=" "0"
            SubStr "[CUR_TIME]" "4" "1" "[CUR_MIN]"
        EndIf

        SubStr "[CUR_TIME]" "6" "2" "[CUR_SEC]"
        SubStr "[CUR_SEC]" "1" "1" "[TVAR]"

        If "[TVAR]" "=" "0"
            SubStr "[CUR_TIME]" "7" "1" "[CUR_SEC]"
        EndIf
    Else
        SubStr "[CUR_TIME]" "4" "2" "[CUR_MIN]"
        SubStr "[CUR_MIN]" "1" "1" "[TVAR]"

        If "[TVAR]" "=" "0"
            SubStr "[CUR_TIME]" "5" "1" "[CUR_MIN]"
        EndIf

        SubStr "[CUR_TIME]" "7" "2" "[CUR_SEC]"
    
```

## DEVELOPING A MULTIMEDIA KIOSK WITH NEOBOOK

```
SubStr "[CUR_SEC]" "1" "1" "[TVAR]"

If "[TVAR]" "=" "0"
    SubStr "[CUR_TIME]" "8" "1" "[CUR_SEC]"
EndIf

EndIf

If "[CUR_MIN]" "=" "[TOTAL_MIN]"
    If "[CUR_SEC]" "=" "[TOTAL_SEC]"
        SetVar "[LOOPS]" "0"
    EndIf
EndIf

EndWhile

. Change the next few lines to execute whatever commands.

StopMedia ""
SetVar "[BUSY]" "NO"
GotoPage "SUB-MENU"
```

## APPENDIX 2

### Specifying individual TIME event with NBW v3.2

Statement lines 1 to 3 are important as it initializes the starting minutes and seconds. Change Line 4 accordingly to execute whatever commands.

```
SetVar "[BUSY]" "YES"
SetVar "[START_MIN]" "[MINUTE]"
SetVar "[START_SEC]" "[SECOND]"
PlayVideoFile "-1" "-1" "-1" "-1" "VIDEO2.MOV" "NORMAL"
```

Specify the minutes and seconds in the following lines:

```
SetVar "[EST_MIN]" "2"
SetVar "[EST_SEC]" "58"
```

Here's where the algorithm starts. You can easily copy/paste it to your NBW publication. More explanation is available in Section 2.

```
Math "60 - [EST_MIN]" "0" "[MIN_REQ]"

If "[START_MIN]" "<" "[MIN_REQ]"
    Math "[START_MIN]+[EST_MIN]" "0" "[TOTALMIN]"
EndIf

If "[START_MIN]" "=" "[MIN_REQ]"
    SetVar "[TOTALMIN]" "0"
EndIf

If "[START_MIN]" ">" "[MIN_REQ]"
    Math "([START_MIN]+[EST_MIN])-60" "0" "[TOTALMIN]"
EndIf

Math "[START_SEC]+[EST_SEC]" "0" "[TOTALSEC]"

If "[TOTALSEC]" ">" "60"
    Math "[TOTALSEC] - 60" "0" "[TOTALSEC]"
    Math "[TOTALMIN] + 1" "0" "[TOTALMIN]"
EndIf

If "[TOTALSEC]" "=" "60"
    SetVar "[TOTALSEC]" "0"
EndIf

SetVar "[LOOPS]" "1"
While "[LOOPS]" "<>" "0"
    SetVar "[CUR_MIN]" "[MINUTE]"
    SetVar "[CUR_SEC]" "[SECOND]"

    If "[CUR_MIN]" "=" "[TOTALMIN]"
        If "[CUR_SEC]" "=" "[TOTALSEC]"
```

## *DEVELOPING A MULTIMEDIA KIOSK WITH NEOBOOK*

```
        SetVar "[LOOPS]" "0"  
    EndIf  
EndIf  
EndWhile
```

. Change the next few lines to execute whatever commands.

```
StopMedia ""  
SetVar "[BUSY]" "NO"  
GotoPage "SUB-MENU"
```

## *Frequently-Asked Questions*

1. When a main program (PARENT.EXE) executes another program (CHILD.EXE), is PARENT.EXE still in memory or only a basic 'shell' code is left so that when CHILD.EXE ends, it will exits back to PARENT.EXE? If this is true, how much memory is used up?

NeoSoft's reply: When a compiled NeoBook publication is run, only a small portion of the file is loaded into memory at any given time. Images, text and media files are cached in and out of memory as they are needed. The amount of actual memory used can vary during the execution of the program and is dependant on which elements are actually in use at the time. NeoBook will also allow Windows 95 to free up memory for use by other programs. I don't think you will have a problem running one publication from another. Unlike DOS, Windows does a good job of providing resources to multiple programs.

2. What is the maximum number of user-definable variables allowed in NBW?

NeoSoft's reply: There is no real limit to the number of user-defined variables in NeoBook. Variables are allocated dynamically and would be limited only by the amount of memory available. With Windows' ability to use virtual memory, you shouldn't have to worry about running out of space for variables. If needed, you can release memory used by a variable, by setting its value to an empty string. For example:

```
SetVar "[MyVariable]" ""
```

3. When a large compiled NBW EXE (say 20MB) is run, it will uses the user-specified TEMP directory to extract certain essential files during execution. Theoretically, how much disk space is required to store these files on the hard disk temporary?

NeoSoft's reply: Files that need to be extracted during execution are video, sound and font files. The temporary space required is theoretically the combined size of all of these files. Image, text, flic (FLI, FLC) and cartoon files are not extracted.