

## 4. Laborator

### 4.1. Scopul laboratorului

Dobândirea cunoștințelor și abilităților teoretice și practice necesare pentru:

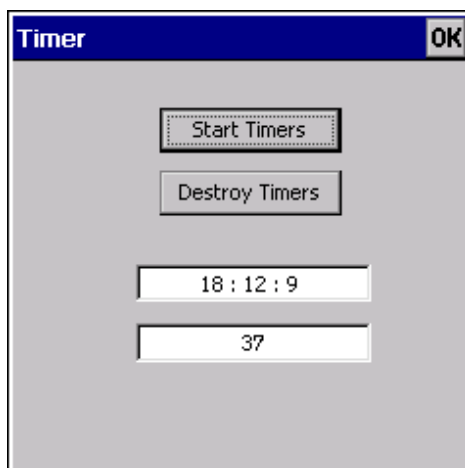
1. utilizarea elementelor de tip timer și
2. tratării diferitelor mesaje ale SO Windows cu particularizare pe Windows Compact Edition.

### 4.2. Cerințe

Dezvoltați un program care să aibă o interfață grafică similară cu cea din

**Figura 4.1.** Programul va fi capabil să realizeze următoarele funcții:

1. Din secundă în secundă să afișeze ora exactă într-un *edit control* – prezentarea orei exacte va fi realizată cu ajutorul unui element de timp *timer*;
2. Din 2.5 secunde în 2.5 secunde să incrementeze cu o unitate și să prezinte într-un *edit control* diferit de cel anterior o valoare numerică care să înceapă, la momentul inițial, cu valoarea 0. Pentru atingerea acestui obiectiv se va folosi un alt element de tip *timer*;
3. Aplicația va avea 2 butoane. Primul buton va porni iar cel de al doilea va opri ambele *timer*-e asociate cu cele două funcții de mai sus;
4. Tratați toate mesajele posibile de erori (de exemplu să nu crez un nou timer daca deja am unul activ etc.)



**Figura 4.1.** Interfața grafică a programului

### 4.3. Timer-e, evenimente și mesaje

*Timer*-ele sunt elemente ce aparțin *kernel*-ului SO și permit declanșarea unor activități în mod periodic. Această perioadă poate fi aleasă de utilizator.

Un *timer* ca orice alt element constitutiv al unui program ce rulează în mediul Windows – precum ferestrele, elementele de tip *button*, *edit control*, *check box*, *progress control* etc. – este capabil să genereze un anumit mesaj. Atunci când orice *timer* se declanșează el va trimite un mesaj [WM\\_TIMER](#) către aplicația noastră. Acest mesaj trebuie tratat în clasa asociată cu Dialog Box-ul care reprezintă tocmai interfața grafică a utilizatorului cu programul nostru.

În mediul IDE Visual Studio vom întâlni noțiunile de **mesaj** și **eveniment**, vezi și **Figura 4.3**.

În cadrul *tab*-ului *Properties* (de ex. a clasei asociate cu interfața grafică a programului) vom găsi listate în rubrica “**Events**” mesajele care sunt trimise de elementele de pe interfața grafică – elementele de tip *controls*. De fapt, cea mai mare parte a acestor „evenimente” prezentate sunt parametrii ce sunt trimiși în cadrul mesajului [WM\\_COMMAND](#). Acestea pot fi asociate drept mesaje de notificare a elementelor de pe interfața grafică (deci, a elementelor de tip *controls*).

În cadrul rubricii „**Messages**” mediul Visual Studio prezintă mesajele trimise de sistemul de operare către aplicația noastră și care pot fi sau nu tratate.


Ce trebuie să reținem este că **mesajele înglobează în ele diferiți parametri; unul dintre acești parametri identifică ce acțiune specifică a fost efectuată cu elementul care a generat mesajul respectiv** – acest parametru este de fapt unul din diferiți identificatori ai acțiunii executate pe element.

De exemplu, când se apasă un buton de pe interfața grafică (apăsarea unui buton este un eveniment care are loc în sistem) se generează mesajul [WM\\_COMMAND](#) acest mesaj va conține drept parametru un identificator a acțiunii efectuată cu respectivul buton, de ex. [BN\\_CLICKED](#) – pe buton s-a dat *click* stânga cu *mouse*-ul (butonul a fost apăsat). În acest moment poate exista un “**message handler**” (o funcție) care să preia mesajul și să execute o anumită acțiune prestabilită sau, din contra, să ignore acest mesaj. Funcția asociată cu mesajul (“**message handler**”) este lansată în execuție în conformitate cu un tabel în care sunt trecute toate mesajele și funcțiile asociate acestora. Mesajul [WM\\_COMMAND](#) este generat de fiecare dată când: un utilizator selectează o opțiune dintr-un meniu, atunci când un element de pe interfața grafică (un *controls*) își schimbă starea în vreun fel sau atunci când o tastă este apăsată.

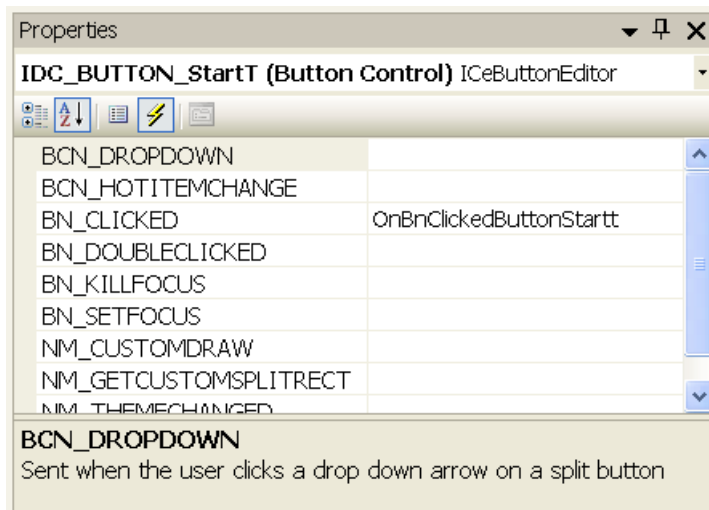
Evenimentele de mai sus nu au nici o legătură cu mecanismul de sincronizare a firelor de execuție prin intermediul evenimentelor (mecanism ce va fi prezentat ulterior în cadrul laboratoarelor), deci nu au nici o legătură cu evenimentele generate sau create de funcții precum [SetEvent](#), [CreateEvent](#), [PulseEvent](#) etc. Deci, acest gen de evenimente (utilizat drept mecanism de sincronizare) nu au nici o legătură cu mesajele – sunt doar noțiuni diferite simbolizate prin același cuvânt.

### 4.4. Lucrul cu timer-e

Toate evenimentele (pe care un element de pe interfața grafică le poate genera) pot fi găsite prin selectarea opțiunii *Properties* din fereastra ce se deschide atunci când se dă click dreapta pe elementul grafic (în *tab*-ul *Resource View* → *Dialog* → *click* pe identificatorul ferestrei principale de interfațare cu utilizatorul → *click* dreapta pe elementul de pe interfața). Deci, aceste evenimente sunt acelea ce pot

fi vizualizate în cadrul panoul *Properties* al elementului grafic apăsând butonul .

Atunci când se utilizează editorului interfeței grafice și se poziționează un nou element pe ea, iar acestui element îi este asociată o funcție (un “*message handler*”), automat se generează codul necesar tratării evenimentului unic în cauză în cadrul acelei funcții – de exemplu, pentru un buton cu ID IDC\_BUTTON\_StartT, atunci când se dă click stânga pe el se va genera evenimentul BN\_CLICKED care este tratat în funcția [OnBnClickedButtonStartt](#), vezi **Figura 4.2**.



**Figura 4.2.** Evenimentele de control asociate cu un element de tip button

După cum am precizat anterior, fiecare timer generează mesajul [WM\\_TIMER](#) periodic, cu o anumită perioadă specificată de utilizator. Dar, la un anumit moment de timp putem avea mai multe *timere* care pot “simultan” să genereze acest mesaj, pentru a le deosebi între ele trebuie să le asociem cu un identificator al acțiunii realizate de fiecare în parte. Deoarece *timer-ul* este un element al *kernel*-ului SO noi nu avem posibilitatea de a-l poziționa în mod grafic pe interfața cu utilizatorul prin intermediul operației *drag and drop* precum celelalte elemente (*button*, *edit control* etc.) și de a-i asocia o funcție. Un element de tip *timer* va fi creat prin intermediul unei funcții. Deoarece noi suntem cei care creăm *timer-ul* tot noi trebuie să definim și identificatorul acțiunii lui. Din acest motiv, definiți un nou identificator global al acțiunii *timer*-ului în programul dumneavoastră:

```
#define WM_TIMER_1S WM_USER + 400
```

SO Windows folosește următorul domeniu de mesaje cu următoarele scopuri:

0	...	WM_USER-1	– Mesaje rezervate pentru utilizare exclusiv de SO.
WM_USER (0x400)	...	0x7FFF	– Definite și utilizate de aplicațiile utilizator pentru a trimite mesaje către clasa asociată cu panoul aplicației (interfața grafică cu utilizatorul). Identificatorii acțiunilor diferitelor elemente pot fi definiți și de către utilizator, cum este și cazul nostru particular, dar având grijă să nu ne suprapunem cu identificatorii pe care aplicația noastră le utilizează deja

		(din acest considerent s-a ales un increment de +400 anterior – presupunând ca nu am mai mult de 400 de mesaje generate de elemente de pe interfața grafică, care sunt tratate în diferite funcții și care prin identificatorii acțiunilor lor proprii să ocupe și să depășească tot spațiul între <b>WM_USER</b> și <b>WM_USER + 400</b> ).
<b>WM_APP</b> ... <b>0xBFFF</b> (0x8000)		– Mesaje cu numere în acest interval sunt disponibile diferitelor aplicații drept mesaje private.
<b>0xC000</b> ... <b>0xFFFF</b>		– Mesaje cu numere în acest interval sunt definite și înregistrate de o aplicație prin intermediul funcției <b>RegisterWindowMessage</b> și sunt mesaje de tip string.
<b>0xFFFF</b> ...		– rezervate de SO;

Mesajul **WM\_USER** este definit în cadrul unui heder a mediului de dezvoltare sub forma: **#define WM\_USER 0x400** (valoare hexa, deci primii 1024 de identificatori sunt rezervați de către SO).

Pentru a crea un element de tip *timer* urmați pașii:

- În fișierul CPP unde sunt implementate funcțiile clasei ce gestionează fereastra de dialog se definește o variabilă de tip **UINT\_PTR Timer1s**. Această variabilă va stoca identificatorul unic al *timer*-ului pe care noi, ulterior, îl vom crea și cu ajutorul căruia îl putem opri, porni sau distruge (opri și elibera resursele asociate *timer*-ului).
- Crearea *timer*-ul se realizează cu ajutorul funcției:

**Timer1s = SetTimer (WM\_TIMER\_1S, „perioadă [milisecunde]”, NULL);**

- De exemplu, pentru distrugerea elementul de tip *timer* utilizați funcția:

**KillTimer (Timer1s);**

**Întrebare:** Ce funcții folosiți pentru a opri sau a porni un *timer* creat anterior?

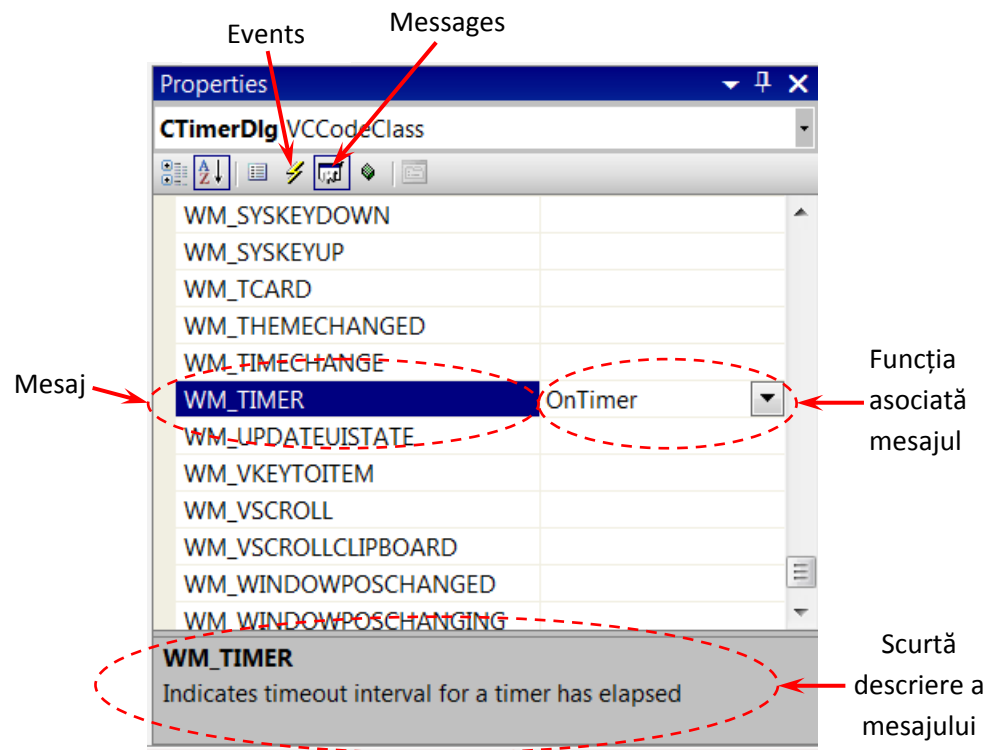
Pentru funcționarea corectă a programului trebuie lămurii următorii trei factori: ce mesaj se generează atunci când un *timer* își finalizează execuția, ce funcție va fi atașată acestui mesaj (evident, cum atașăm această funcție) și în ce clasă se va găsi această funcție care va trata mesajul.

Pentru a prelua mesajul generat de un anumit *timer*, după cum s-a precizat și anterior, se tratează mesajul generic **WM\_TIMER**. De fiecare dată când orice *timer* își finalizează perioada (a trecut timpul precizat în cadrul funcției **SetTimer**) se generează mesajul **WM\_TIMER**.

Tratarea oricărui mesaj, deci și a mesajului **WM\_TIMER**, se realizează prin asocierea acestuia cu o funcție (“*message handler*”), deci prin precizarea numelui funcției ce va fi apelate atunci când se generează mesajul. Pentru atingerea acestui obiectiv se realizează pașii:

- În tab-ul *Class View* se selectează clasa care va prelua mesajul dorit – în cazul nostru, clasa asociată cu **Dialog Box**-ul – fereastra principală a programului;

- b. Prin selectarea acestei clase în fereastra *Properties*, asociată în mod direct și unic clasei, se vor prezenta toate evenimentele și mesajele unice pe care doar această clasă le poate trata;
- c. În continuare, în fereastra *Properties* se apasă butonul *Messages*. În acest moment toate mesajele ce pot fi tratate de clasa selectată vor fi afișate, vezi **Figura 4.3**;
- d. Se selectează mesajul pe care dorim să îl tratăm – în cazul nostru particular **WM\_TIMER**;
- e. Se asociază numele funcției (nume predefinit de mediul de dezvoltare) cu mesajul pe care dorim să îl trateze clasa.



**Figura 4.3.** Interfața grafică în care se realizează asocierea dintre un mesaj și funcția ce va fi apelată

În cazul existenței mai multor *timere* diferențierea între acestea (codul particular ce trebuie executat) se realizează prin identificatorul **nIDEvent** trimis ca argument, de către sistemul de operare, funcției ce tratează mesajul **WM\_TIMER** – în cazul nostru particular, vezi **Figura 4.3**, aceasta va fi **OnTimer**.

Prin intermediul unei construcții de tip **Switch** sau **If** având drept argument **nIDEvent** se identifică *timer*-ul care a generat mesajul, în cazul nostru particular va fi **WM\_TIMER\_1S**, și se asociază codul necesar a fi executat.

Pentru aflarea orei exacte:

- a. Pentru preluarea orei exacte folosiți-vă de o variabilă de tip clasă **CTime**:

```
CTime ct=CTime::GetCurrentTime();
```

- b. Ulterior utilizați funcțiile `GetHour()`, `GetMinute()`, `GetSecond()`, funcții interne clasei `CTime` pentru a prelua informațiile ce țin de ora, minutul și secunda pe care doriți să le afișați de la variabila `ct`.

#### 4.5. Exerciții

1. Introduceți codul necesar generării unui sunet și afișării unui *message box* de avertizare atunci când închideți aplicația - dând totodată utilizatorului posibilitatea să o închidă sau să continue lucrul în ea. Informații ajutătoare:
  - a. Ce mesaj se generează atunci când se închide o aplicație ?
  - b. Pentru generarea unui sunet utilizați:

`MessageBeep(0xFFFFFFFF);`

2. Realizați selectarea celor 2 timere în funcția asociată tratării mesajului `WM_TIMER` cu ajutorul unei structuri de tip `Switch`.
3. Imediat după apăsarea butonului “**Start Timers**” invalidați-l pe acesta pentru a nu mai crea din nou alte *timere*. Butonul va fi revalidat după apăsarea butonului “**Destroy Timers**”.
4. Validați/invalidați în mod similar butonul “**Destroy Timers**” astfel încât să nu se permită distrugerea unor *timere* inexistente.
5. Modificați afișarea orei astfel încât atunci când ora, minutul sau secunda este formată dintr-un singur digit automat să se introducă cifra zero în față.