Der SpDrS60-Timer ist ein Objekt, das als einfacher Signalgeber für zeitgesteuerte Ereignisse als auch zum zeitabhängigen Funktionsaufruf genutzt werden kann.

Es besteht die Möglichkeit, innerhalb der Timerlaufzeit "*Events*" zu definieren. Für die Timer-Events kann ein einzelnes als auch ein Block von Events als Parameter übergeben werden. Diese Events werden dann zum angegebenen Zeitpunkt ausgeführt.

Die grundlegende Konfiguration der Tabelle erfolgt nach folgendem Muster:

used_Timer_SpDr = { {Datensatz1}, {Datensatz..n} }

Die Tabelle wird zur Definition verwendet. Mehrere Datensätze in der Tabelle werden durch ein Komma voneinander getrennt.

Ein Datensatz in der Tabelle ***kann*** die folgenden Parameter enthalten (Beispiel):

{ ObjektId= "Timer01" , SecRuntime= 45, TexTid= "#3", MemoId = "#41", BtnId= 4, DtcId= 2, DtcActivePos= 3 }

ObjektId = "MeinTimer01"
Bezeichner des Timers. Darf nur einmal in der Tabelle ***used_Timer_SpDr*** vorkommen. Der Datentyp ist ein *String*.

**SecRuntime = 45*
Die Timer-Gesamtlaufzeit in Sekunden. Der Datentyp ist eine *Ganzzahl*.

TexTid = "#3"
Identnummer des Ausgabeobjekts (z.B. *SpDrS60_TexT_NmFld_kl*) zur Anzeige der verbleibenden Laufzeit in Sekunden. Der Datentyp ist ein Immo-Identstring. Das Ausgabeobjekt muss das Feature *"Texturentext"* unterstützen.

MemoId = "#41"
Identstring des Immo-Objekts bei Verwendung von TagText als Speichermedium für die Echtzeitdaten des Timers. Der Datentyp ist eine Immo-Identnummer als String.

BtnId = 4
Identnummer des Starttasters, falls installiert. Der Datentyp ist eine *Ganzzahl*.

DtcId = 2
Identnummer der Melder-LED, falls installiert. Signalisiert den Zustand des Timers. Der Datentyp ist eine *Ganzzahl*.

DtcActivePos = 3
Aktive Signalposition der Melder-LED, falls eine installiert ist. Ändert den Standardwert 2 (Dauerlicht) der SpDr-LED in die Position 3 (Blinklicht). Der Datentyp ist eine Ganzzahl.

(Mit einem ***** markierte Parameter sind Pflichtparameter)

Die Methoden des Timerobjekts

Durch die Definition eines Timers in der Tabelle wird in der SpDrS60-Umgebung ein Timerobjekt sowie in der Textdatei *SpDrS60_TmpFunctionFile.txt* Methodeneinträge generiert. Um eine eindeutige Zuordnung zu gewährleisten, werden die Funktionseinträge mit der Timer-ID des Datensatzes initialisiert. Hier ein Beispielausschnitt aus der *TmpFunctionFile.txt*.

```
------ Methoden fuer Timer -- MeinTimer01 -------------
MeinTimer01_Start ------------ Aufruffunktion - Timerstart ueber Kontakt -
MeinTimer01.TimerStart()
MeinTimer01.getState()
-----------------------------------------------------------------------
```

Alle folgenden Beschreibungen beziehen sich auf den Beispieleintrag " MeinTimer01".

MeinTimer01.TimerStart()

startet den Timer.

MeinTimer01.TimerStop()

stoppt den Timer.

MeinTimer01.getState()

liefert den aktuellen Zustand des Timers als Boolean. Lauf = *true* / Stop = *false*

MeinTimer01.setRunTime(30)

initialisiert die Timerlaufzeit, ***überschreibt*** **den Altwert!**

MeinTimer01.setTexTid("#14")

initialisiert ein Immo-Ausgabeobjekt zur Anzeige der verbleibenden Sekunden. *Das Ausgabeobjekt muss das Feature "Texturentext" unterstützen.* Das SpDrS60-Inventar enthält die entsprechenden Bauteile. **Die Methode überschreibt einen bestehenden Altwert!**

MeinTimer01.setEvent(10,"MyFunc(x,y)")

übergibt ein Einzelevent an den Eventcontroller des Timers.

MeinTimer01.setEventTbl(MyEventTable)

übergibt eine Eventtable an den Eventcontroller des Timers.

MeinTimer01.clearEvents()

Entleert den Eventspeicher des Timers.

Die Events

Beim Eventspeicher handelt es sich um einen Bereich, der durch den Timer zyklisch abgefragt wird. Über die o.g. Methoden lässt sich der Speicher verwalten. Hier Beispiele:

MeinTimer.setEvent(10,"EEPSetSignal(8,2)")

Mit der Methode .setEvent() lässt sich ein einzelnes Ereignis an das Timerobjekt übergeben. Als ersten Parameter erwartet die Funktion den Zeitpunkt der Ausführung in Sekunden nach Timerstart (hier 10 Sek.) und das auszuführende Event als String. (hier: *"EEPSetSignal(8,2)")*

MeinTimer.setEventTbl(MyEventCollection)

Mit der Methode .setEventTbl() lässt sich die Referenz auf eine Sammlung von Ereignissen an das Timerobjekt übergeben. Als Parameter erwartet die Funktion den Bezeichner einer Tabelle (im Bsp. *MyEventCollection)*.

Die Tabelle muss dazu in folgender Form erstellt werden:

```
MyEventCollection = {
        {8,"EEPSetSwitch(24,1)"},
        {10,"EEPSetSignal(8,2)"},
        {10,"EEPSetSignal(12,2)"},
};
```

Danach erfolgt die Übergabe:

MeinTimer01.setEventTbl(MyEventCollection);

Wie man an dem Beispiel erkennen kann, handelt es sich syntaktisch um die gleichen Parametertypen und deren Reihenfolge wie bei der Zuweisung als einzelnes Ereignis in der setEvent-Methode. Jetzt wurde jedoch eine Befehlssequenz als Tabelle gekapselt und mit der entsprechenden Funktion übergeben.

Das Timer-Speichermodell

Um den aktuellen Zustand des Timers bei Anlagen-Neustart oder "Script neu laden" durch den EEP-Luaeditor wieder herzustellen, benötigen das Objekt einen Speicherplatz für die relevanten Daten. In der Regel erfolgt das Speichern der Daten automatisch in einem Datenslot, der einmalig bei Initialisierung des Timers durch einen Slotmanager reserviert wurde.

Der Timerslot bleibt bis zum Entfernen der Timerdefinition aus der Table *"used_Timer_SpDr"* erhalten. Die Speichermethoden des Timers greifen also immer auf die gleiche Slotnummer zu.

Seit EEP15 besteht die Möglichkeit, bei Immobilien und Rollmaterialien Daten in sog. *"Tag-Texten"* zu hinterlegen und mit der Anlage speichern zu lassen. Das entlastet das auf 1000 Slots begrenzte Kontingent und ermöglicht so ein alternatives Speichermedium für das Objekt.

Der SpDrS60-Timer nutzt diese Technologie auf einfache Art und Weise. Wird im Datensatz der Parameter *"Memold"* mit einem gültigen Immobilien-Identstring versorgt, verwendet das Objekt automatisch die TagText-Funktion zum Speichern der Echtzeitdaten.

Sollte der Timer bisher mit einem Speicherslot betrieben worden sein, wird sein Inhalt in den TagText der Immobile übernommen und der entsprechende Slot zur weiteren Verwendung wieder freigegeben.