Die Bibliothek SysLibCallback.lib

Diese Bibliothek enthält die Funktionen **SysCallbackRegister** und **SysCallbackUnregister**, die dazu dienen, definierte Callback-Funktionen für Laufzeitereignisse zu aktivieren.

Änderungen für SysLibSockets23.library sind blau markiert.

Beide Funktionen sind vom Typ BOOL und liefern jeweils TRUE, wenn die angegebene Callback-Funktion registriert bzw. deregistriert werden kann.

Der Prototyp der Callback-Funktion muss wie folgt aussehen:

```
FUNCTION Callback : DWORD
VAR_INPUT
    dwEvent:DWORD;   (* Event *)
    dwFilter:DWORD;  (* Filter *)
    dwOwner:DWORD;   (* Source *)
END_VAR
```

Achtung für RISC und Motorola 68K Zielsysteme: Der Name der Callback-Funktion **muss** mit „callback“ beginnen!

Die Bibliotheksfunktionen **SysCallbackRegister** und **SysCallbackUnregister** verwenden jeweils die folgenden Übergabeparameter beim Ansprechen der zu registrierenden/deregistrierenden Callback-Funktion:

Input-Variable	Datentyp	Beschreibung
iPOUIndex	INT	POU Index der zu (de)registrierenden Callback Funktion. Dieser wird vorher mit Hilfe des Operators **INDEXOF**(<Funktionsname>) ermittelt.
Event	RTS_EVENT	Das Laufzeit-Ereignis, für das die Callback-Funktion verwendet werden soll, wird durch einen Wert der Enumeration RTS_EVENT beschrieben, die ebenfalls in der Bibliothek enthalten ist (siehe unten) **Achtung:** In SysLibCallback23.library werden nicht alle der unten angeführten Events unterstützt!

Die Enumeration RTS_EVENT ist folgendermaßen definiert; in SysLibCallback23.library nicht unterstützte Events sind blau markiert:

```
TYPE RTS_EVENT :
(
    EVENT_ALL,
    (* General events *)
    EVENT_START,
    EVENT_STOP,
    EVENT_BEFORE_RESET,
    EVENT_AFTER_RESET,
    EVENT_SHUTDOWN,
```

```
(* Exceptions generated by runtime *)
EVENT_EXCPT_CYCLETIME_OVERFLOW,          (* Cycle time overflow *)
EVENT_EXCPT_WATCHDOG,                    (* Software watchdog OF IEC-task expired *)
EVENT_EXCPT_HARDWARE_WATCHDOG,           (* Hardware watchdog expired. Global software error *)
EVENT_EXCPT_FIELDBUS,                    (* Fieldbus error occurred *)
EVENT_EXCPT_IOUPDATE,                    (* IO-update error *)

(* Exceptions generated BY system *)
EVENT_EXCPT_ILLEGAL_INSTRUCTION,         (* Illegal instruction *)
EVENT_EXCPT_ACCESS_VIOLATION,            (* Access violation *)
EVENT_EXCPT_PRIV_INSTRUCTION,            (* Privileged instruction *)
EVENT_EXCPT_IN_PAGE_ERROR,               (* Page fault *)
EVENT_EXCPT_STACK_OVERFLOW,              (* Stack overflow *)
EVENT_EXCPT_MISALIGNMENT,                (* Datatype misalignment *)
EVENT_EXCPT_ARRAYBOUNDS,                 (* ARRAY bounds exceeded *)
EVENT_EXCPT_DIVIDEBYZERO,                (* Division BY zero *)
EVENT_EXCPT_OVERFLOW,                    (* Overflow *)
EVENT_EXCPT_NONCONTINUABLE,              (* Non continuable *)
EVENT_EXCPT_NO_FPU_AVAILABLE,            (* FPU: No FPU available *)
EVENT_EXCPT_FPU_ERROR,                   (* FPU: Unspecified error *)
EVENT_EXCPT_FPU_DENORMAL_OPERAND,        (* FPU: Denormal operand *)
EVENT_EXCPT_FPU_DIVIDEBYZERO,            (* FPU: Division BY zero *)
EVENT_EXCPT_FPU_INVALID_OPERATION,       (* FPU: Invalid operation *)
EVENT_EXCPT_FPU_OVERFLOW,                (* FPU: Overflow *)
EVENT_EXCPT_FPU_STACK_CHECK,             (* FPU: Stack check *)

(* IO events *)
EVENT_AFTER_READING_INPUTS,
EVENT_BEFORE_WRITING_OUTPUTS,

(* Miscellaneous events *)
EVENT_TIMER,                             (* Schedule tick (timer interrupt) *)
EVENT_DEBUG_LOOP,                        (* Debug loop at breakpoint *)
EVENT_SCHEDULE,                          (* Schedule tick (timer interrupt), is called always instead of EVENT_TIMER *)
EVENT_ONLINE_CHANGE,                     (* Is called after CodeInit() at Online-Change *)
EVENT_BEFORE_DOWNLOAD,                   (* Is called at the beginning of a program download *)
```

```
EVENT_TASKCODE_NOT_CALLED,          (*Is called in cyclic task, if IecCode is NOT called (instead of).*)
EVENT_SYNC_RECEIVED,
EVENT_BEFORE_READING_INPUTS,
EVENT_AFTER_WRITING_OUTPUTS,
EVENT_SYSTEM_CRASH,
EVENT_POWERFAIL,
EVENT_CANMESSAGE_RECEIVED,          (*The parameter source in IEC is the device index, the parameter filter is the pointer to the message.*)
EVENT_EXCPT_PLC_OVERLOAD,
EVENT_TARGETVISU_WINDOW_ACTIVATED,
EVENT_TARGETVISU_WINDOW_DEACTIVATED,
EVENT_BEFORE_ONLINE_CHANGE,         (* it's possible to release resources here which block a task so that OLC could not be started *)
EVENT_AFTER_ONLINE_CHANGE,          (* code execution has switched to the new code completely *)

(* Online services *)
EVENT_ONLINE_SERVICES_BEGIN := 500,
EVENT_LOGIN,
EVENT_CUSTOM_SERVICES,

(* Interrupts *)
EVENT_INT_0:=1000,
EVENT_INT_1,
EVENT_INT_2,
EVENT_INT_3,
EVENT_INT_4,
EVENT_INT_5,
EVENT_INT_6,
EVENT_INT_7,
EVENT_INT_8,
EVENT_INT_9,
EVENT_INT_10,
EVENT_INT_11,
EVENT_INT_12,
EVENT_INT_13,
EVENT_INT_14,
EVENT_INT_15,
```

```
        EVENT_INT_255:=1255,

        EVENT_MAX
);
END_TYPE
```