Programmpaket

convert_lib

Konvertierung von Simatic-Formaten <-> C-Formaten

Version V2.7 (10.05.2018)

B E S C H R E I B U N G

convert_lib wurde entwickelt von

Heisch Automation

Ingenieurbüro für Industrieautomatisierung

Adresse:	Im Vorderen Großthal 4	D 76857 Albersweiler /Pfalz
Tel:	+49 6345 9496732	Mobil: +49 171 4311359
e-mail:	sales@heisch-automation.de	www.heisch-automation.de

http://sites.inka.de/heisch
http://www.heisch-automation.de

Inhaltsverzeichnis

1. ALLGEMEINES

convert_lib beinhaltet Funktionen zur Umrechnung von Simatic-Datenformaten zu Rechner-Formaten und zurück.

convert_lib unterstützt sowohl Simatic S5 als auch Simatic S7 300 / 400 und 1200 / 1500.

convert_lib unterstützt die Datenformate, die zum Datenaustausch benutzt werden, Pointer-Formate werden nicht unterstützt.

convert_lib ist ein add on zu der rk*_server-Familie, kann aber völlig unabhängig davon benutzt werden.

1.1. Warenzeichen

RK512, 3964R, SIMATIC sind Warenzeichen der SIEMENS AG.
S.u.S.E. Linux ist ein Warenzeichen der S.u.S.E. GmbH.
UNIX ist ein Warenzeichen der X/Open Company Limited.
MS-Windows ist ein Warenzeichen der Microsoft Corporation.

1.2. Unterstützte Betriebssysteme und Systemanforderungen

convert_lib wurde beginnend mit S.u.S.E-Linux 6.3 entwickelt, sollte aber auch auf jeder anderen Linux-Distribution laufen, sofern sie auf der glibc basiert.

Weil convert_lib in Quellcode ausgeliefert wird, sollte es zusätzlich auch auf jedem anderen Unix-System laufen.

Momentan unterstützen wir MS-Windows nicht, aber wir sind recht sicher, daß sich die meisten Funktionen mit sehr geringem Aufwand auch unter MS-Windows nutzen lassen.
Ausnahmen: Die Datum-Uhrzeit-Funktionen sind sehr Unix-spezifisch.

(Ein Kunde nutzt u.a. unsere Gleitpunktfunktionen für den Zugriff aus Simatic S7 in C-Programmen, compiliert mit den Borland C-Builder.)

1.3. Änderungen (WICHTIG für upgrades)

Ab Version 2.7

Der ursprüngliche Ansatz war, die Funktionen soweit aufzuteilen, dass möglichst nur die benötigten Funktionen mit einkompiliert werden müssen. (Ein gut ausgestatteter Linux-Rechner hatte damals ca. 64 MB) Die Funktionen wurden nun alle in eine Datei zusammengelegt.

Da Simatic S5 Steuerungen immer mehr obsolet werden, mit S7-1200 und S7-1500 aber neue Formate hinzukamen wurden die Funktionen in 2 Bereiche aufgeteilt:

- S5 : die S5-Formate bleiben in s5types.h, S5types.c und s5types.o
  Die Datei 's5types.c' beinhaltet den letzten Stand von 2016, aus Rückwarts-Kompatibilitäts-Gründen sind alle, auch die s7-spezifischen Funktionen enthalten.
- S7 : die S7-Formate sind jetzt in s7types.h, S7types.c und s7types.o
  Für Neuentwicklungen sollte nur noch dieser Zweig benutzt werden.
  Die Funktionsnamen des S7-Bereichs wurden stärker an die Namen der Variablentypen der S7 angenähert, z.B. Lesen einer INT-Variablen : get_s7INT().
  Um diese Bibliothek zwischen verschiedenene Architekturen maschinenunabhängiger zu machen, werden jetzt die in stdint.h definierten Integer-Funktionen (int8_t, uint16_r, etc) benutzt.

Nicht aufgenommen wurden Funktionen zum Lesen und Schreiben von WCHAR und WSTRING.

Ab Version 2.0

Mit dem Wechsel der Versionsnummer auf V2.0 haben wir die Parameterübergabe der Simatic-bezogenen Parameter von **unsigned short *in** zu **void *in** und **unsigned short *out** zu **void *out** geändert.

Grund:
Die ersten Versionen von convert_lib wurden für die Simatic S5 entwickelt.
Ein Wort in der Simatic S5 besteht aus 16 Bits. Um den Zusammenhang Simatic-Wort zu Variable im Rechner transparent zu halten, benutzen viele Kunden Konstrukte wie array of unsigned short.

Zum Beispiel: DB20, DW 0 bis DW 48 wird in ein "unsigned short db20[50] "gelesen.
Daraus ergibt sich sofort: Der Inhalt des DW 38 steht in der Variable db20[38].

Seit Simatic S5 am aussterben ist und nahezu die meisten Kunden Simatic S7 benutzen, ist das Parameterformat unsigned short *in kontraproduktiv, weil es je Aufruf eine cast-Operation benötigt.

Wir haben uns deshalb entschlossen, die Parameter als "void *in" zu übergeben., dies ermöglicht die größte Flexibilität.

1.4 Lieferumfang

Das Programmpaket besteht aus den Dateien

s5types.h	das header file S5-Funktionen
s5types.c	Alle S5 Funktionen zusammen.
s7types.h	das header file S7-Funktionen
s7types.c	Alle S7 Funktionen zusammen.
Makefile	das Makefile
Changes.txt	aktuelles
Manual.pdf	Manual as Pdf-file, English
Beschreibung.pdf	Beschreibung als PDF-Datei, deutsch.

1.5 Fehlende Funktionen und Upgrades

Nicht aufgenommen wurden Funktionen zum Lesen und Schreiben von WCHAR und WSTRING.

Altkunden bekommen den jeweils neuesten Stand der convert_lib kostenlos per e-mail. Daraus kann aber keine Lieferverpflichtung unsererseits abgeleitet werden.

1.6. Haftungsbeschränkung

Bei der convert_lib handelt es sich um eine im betrieblichen Einsatz erprobte Software.
Trotz der bisher gezeigten Zuverlässigkeit kann jedoch nach dem Stand der Technik eine vollständige Fehlerfreiheit nicht garantiert werden.

Bitte beachten Sie deshalb:
Die Benutzung der vorliegenden Software erfolgt auf eigene Gefahr! In keinem Fall wird für Schäden eine Haftung übernommen, die direkt oder indirekt durch Anwendung der Software verursacht werden.

2. s7types.c, s7types.h FUNKTIONEN

2.1 Ganzzahl-Formate

Byte-Funktion sind eigentlich zu trivial, sie wurden nur aus Gründen der Einheitlichkeit aufgenommen.

get_s7BYTE (*source)

S7: importiert ein DBB von Typ Byte (uint8_t)

uint8_t get_s7BYTE (void *in) ;

set_s7BYTE (*dest,source)

Schreiben eines unsigned 8-Bit-Werts zu einem DBB

void set_s7BYTE (void *out, unsigned char in) ;

get_s7USINT (*source)

S7: importiert ein DBB von Typ unsigned int8 (unsigned char) als (uint8_t)

uint8_t get_s7USINT (void *in) ;

set_s7USINT (*dest,quelle)

Schreiben eines unsigned 8-Bit-Werts als USINT

void set_s7USINT (void *out, unsigned char in) ;

get_s7CHAR (*source)

S7: importiert ein DBB von Typ char (unsigned char) als (unsigned char)

signed char get_s7CHAR (void *in) ;

set_s7CHAR(*dest,source)

Schreiben eines signed 8-Bit-Werts zu einem DBB

void set_s7CHAR (void *out, signed char in) ;

get_s7SINT (*source)

S7: importiert ein DBB von Typ SINT signed 8bit integer

int8_t get_s7SINT (void *in) ;

set_s7SINT (*dest,source)

Schreiben eines signed 8-Bit-Werts als SINT

void set_s7SINT (void *out, int8_t in) ;

get_s7WORD (*source)

importiert ein 16 bit wort als unsigned int 16

uint16_t get_s7WORD (void *in) ;

set_s7WORD (*dest,source)

Schreiben eines unsigned 16-Bit-Worts als WORD

void set_s7WORD (void *out,uint16_t in) ;

get_s7UINT (*source)

importiert ein 16 bit Wort als unsigned int 16

uint16_t get_s7UINT (void *in) ;

set_s7UINT (*dest,source)

Schreiben signed 16-Bit-Worts als INT

void set_s7UINT (void *out,uint16_t in) ;

get_s7INT (*source)

importiert eine 16 bit signed-Zahl

int16_t get_s7INT (void *in) ;

set_s7INT (*dest,source)

Schreiben eines signed 16-Bit-Worts als INT

void set_s7INT (void *out, int16_t in) ;

get_s7DWORD (*source)

Doppelwort als 32-Bit unsigned Integer einlesen
importiert eine unsigned Zahl

uint32_t get_s7DWORD(void *in) ;

set_s7DWORD (*dest,source)

Schreiben eines unsigned 32-Bit-Worts als DWORD

void set_s7DWORD (void *out,int in) ;

get_s7UDINT (*source)

Doppelwort als unsigned long einlesen
importiert eine unsigned Zahl

uint32_t get_s7UDINT(void *in) ;

set_s7UDINT (*dest,source)

Schreiben eines unsigned 32-Bit-Worts als UDINT

void set_s7UDINT (void *out,int in) ;

get_s7DINT (*source)

Doppelwort als long integer einlesen
importiert eine signed-Zahl

int32_t get_s7DINT(void *in) ;

set_s7DINT (*dest,source)

Schreiben eines signed 32-Bit-Worts als DINT

void set_s7DINT (void *out,int in) ;

get_s7LWORD(*source)

Longwort (64 bit) als unsigned long einlesen
importiert eine 64 bit unsigned Zahl
S7-1200 / s7-1500

uint64_t get_s7LWORD(void *in) ;

set_s7LWORD (*dest,source)

Schreiben eines unsigned 64-Bit-Worts als LWORD

void set_s7LWORD (void *out, uint64_t in) ;

get_s7ULINT(*source)

importiert eine 64 bit unsigned Zahl
S7-1200 / s7-1500

uint64_t get_s7ULINT(void *in) ;

set_s7ULINT(*dest,source)

Schreiben eines unsigned 64-Bit-Worts als ULINT

void set_s7ULINT (void *out, uint64_t in) ;

get_s7LINT (*source)

Long Integer (64 bit) als long integer einlesen
S7-1200 / s7-1500

int64_t get_s7LINT(void *in) ;

set_s7LINT(*dest,source)

Schreiben eines signed 64-Bit-Worts als LINT

void set_s7LINT (void *out, int64_t in) ;

get_s7COUNTER (*source)

importiert einen Zählerwert (BCD 0..999) als Integer

int get_s7COUNTER (void *in) ;

set_s7COUNTER (*dest, source)

Schreiben eines Zaehlerwerts
Returncode:
0 = OK
-4 = Wert < 0. ausser Bereich
-5 = Wert > 999 : ausser Bereich

int set_s7COUNTER (void *out, int in) ;

2.2 Gleitpunktformate

get_s7REAL(*source)

S7-REAL-Zahl als double einlesen

Format in S7:
Bit 31 = SIGN
Bit 30..23 = 8 Bit Exponent
Bit 22.. 0 = 23 Bit Mantisse

double get_s7REAL(void *in) ;

set_s7REAL (*dest,source)

Schreiben einer double zu einer S7-REAL-Zahl

returncode : 0 = OK;
-6 = Ausser Bereich : > 3.402823E+38
Die Zahl wäre in der Simatic nicht mehr darstellbar, sie wird auf 3.402823E+38 begrenzt.
-7 = Ausser Bereich : < 1.175495E-38
Die Zahl wäre in der Simatic nicht mehr darstellbar, sie wird als 0.0 übergeben.

int set_s7REAL (void *out, double source) ;

get_s7LREAL(*source)

S7-LREAL-Zahl (64 Bit-) als double einlesen

Format in S7:
Bit 63 = SIGN
Bit 62..52 = 11 Bit Exponent
Bit 51.. 0 = 52 Bit Mantisse

double get_s7LREAL(void *in) ;

set_s7LREAL (*dest,source)

Schreiben einer double zu einer S7-LREAL-Zahl (64 Bit-)

returncode : 0 = OK;
-8 = Ausser Bereich : > 1.7976931348623158e+308
Die Zahl wäre in der Simatic nicht mehr darstellbar, sie wird auf 1.7976931348623158e+308 begrenzt.
-9 = Ausser Bereich : < 2.2250738585072014e-308

Die Zahl wäre in der Simatic nicht mehr darstellbar, sie wird als 0.0 übergeben.

limits according to IEEE754
-1,7976931348623158e+308 to 2,2250738585072014e-308

int set_s7LREAL (void *out, double val) ;

2.3 S7-String-Formate

S7-String Format: <ssize><length>< body ...>

get_s7STRING (*source)

das Simatic-S7-STRING "*in" in ein string dest[] einlesen, maximale Länge des Zielstrings = dmax; Der übernommene String wird mit '\0' abgeschlossen.
Der Zielstring im Rechner muss mindestens einen Character länger als der <body> des Strings in der Simatic sein, denn er muss noch die Endekennung '\0' aufnehmen können.

return code : >= 0 : Anzahl der gelesenene Zeichen in dstr[]
= -10 : Fehler: in S7: <ssize> kleiner als <length>:
Längenangaben in S7 nicht plausibel.
<length> bytes gelesen
Länge von dest[] mit strlen(dest) ermitteln.
: = -11 : Fehler: String *in länger als dest:
Zielstring dest[] zu kurz.
nur dmax -1 bytes gelesen

int get_s7STRING (void *in, char dstr[],int dmax);

get_s7STRING_SIZE (*source, int *length)

Size und Length des S7-Strings "*in" lesen
Der String-Inhalt (= <body> wird nicht gelesen,
nur die Verwaltungs- Einträge Max-Länge (= Size)
und aktuelle Länge (= length)

return code : >= 0 : Size
Wenn 'int *length' != NULL dann auch Übernahme der Länge
des aktuellen Inhalts.

int get_s7STRING_SIZE (void *in, int *length) ;

set_s7STRING ()

Den String src[] in den Simatic-S7-STRING "*out" schreiben,
maximale Länge des Zielstrings = smax;
!! smax muß mit der Datendeklaration im S7-DB übereinstimmen,
!! z.B: in DB: STRING[5] -> smax = 5.
return code : = 0 : kein Fehler
= -12 : Fehler, ungültige Länge smax: gültige Werte für smax = 1.. 254
Es werden ersatzweise für <ssize> 254 Bytes geschrieben.
= -13 : Warnung: Quelle zu lang für Zielstring, wird gekürzt übertragen.

int set_s7STRING (void *out,const char src[],int smax) ;

2.4 Datums- und Uhrzeit-Formate

Stand / Version ab 05.04.2017
- neue Funktionsnamen und neue Funktionen.
- Die alten functionen sind in "s7types_t_obsolete.c".

Die neuen Funktionen beinhalten keine Zeitzonenumrechnung mehr. sie nehmen die Zeit einfach wie sie kommt.
Grund:
- Die neuen CPUs (1200/1500) beinhalten bereits im Betriebssystem die Möglichkeit, zwischen UTC (Systemzeit) und Lokalzeit zu unterscheiden.
- Auch für "Classic"-CPUs besitzen mittlerweile Bibliotheksfunktionen, die die Umrechnung bereitstellen.
- Uhrzeit und Datum unterliegen historischen und politischen Vorgaben, eine korrekte Umsetzung kann beliebig komplex ausfallen, ist aber in den meisten Fällen obsolet, da die Bezugszeit ohnehin Lokalzeit ist.

Um eine möglichst einheitliche Schnittstelle zu den Datums-und Zeit-Funktionen zu ermöglichen, wurde eine **neue Structure 'struct s7hTime'** eingeführt, die im Gegensatz zu den gängigen Unix / Linux - Zeit-Structures sowohl die "broken down date and time" als auch Einheiten von Sekundenbruchteilen enthält.

Zwei neue Funktionen read_s7TimeStr () und strfs7Time() erleichtern die Umrechnung von (Rechner-seitigen) ASCII-Strings zu **'struct s7hTime'** und umgekehrt.

Zur Umrechnungen zwischen 'struct tm', 'struct timeval', 'struct timespec' und 'struct s7hTime' wurden neue Funktionen eingeführt.

Als zusätzlichen Eigenschaften beinhalten die neuen Funktionen die Möglichkeit, die Zeiten zusätzlich in den Formaten 'struct timespec oder 'struct tm' zu übergeben oder zu übernehmen.
Dazu muss jeweils die gewünschte Structure vom Benutzer angelegt und als Parameter ubergeben werden.

Die früher verschiedentlich benutzte 'struct timeval' wurde durch 'struct timespec' ersetzt, da 'timespec' auch Nanosekunden übermitteln kann. Dadurch wird die Beschaltung der Funktionen einheitlich.

Alle Funktionen haben einen zusätzlichen Parameter, über den Nanosekunden übergeben werden. Der Parameter beinhaltet die Bruchteile von Sekunden in ns-Einheiten.

Ausnahme: get_DATE(),. set_DATE()

Für die zur Simatic schreibenden Funktionen (set_s7***) gilt:
Alle Format-Pointer-Parameter werden von links nach rechts abgearbeitet. Der Letzte Nicht-NULL gewinnt.

! 'struct tm' kann keine Sekunden-Bruchteile übergeben.
Dafür wird der Wert aus 'ns' oder, falls definiert, ts->tv_nsec' genommen.

Wo sind die 'classischen' Unix-/Linux- Zeit-Structures dokumentiert ?

/usr/include/sys/time.h:

```
struct timespec
{
 time_t tv_sec;                  // Seconds.
 long int tv_nsec;          // Nanoseconds
};
```

Für struct tm: "man gmtime":

```
  struct tm {
      int tm_sec;         /* seconds
      int tm_min;         /* minutes
      int tm_hour;        /* hours
      int tm_mday;        /* day of the month
      int tm_mon;         /* month (0..11)
      int tm_year;        /* year (1900 = 0 ...)
      int tm_wday;        /* day of the week (0..6) (*)
      int tm_yday;        /* day in the year (0 = 1.Jan .. 365) (*)
      int tm_isdst;       /* daylight saving time
  };
```

/* ---------- time structure for convertlib -------- */

typedef struct s7hTime : Neue Struktur für Zeit-Darstellung

Diese structure ist ein zusätzliches Zeit-Darstellungsformat für die 'convertlib'-Bibliothek.

Sie ist angelehnt an die 'struct tm' (siehe: /usr/include/time.h) beinhaltet aber zusätzliche Informationen die insbesondere für die Konversion von Datum und Zeit zwischen String-Darstellung und S7-Zeitformaten hilfreich sind.

Bei der Benutzung zusammen mit den string-Funktionen werden zusätzlich die Arten und Positionen von Fehlern hinterlegt.

Bei Benutzung zusammen mit den convert-lib - Konversionsfunktionen werden ebenfalls einige error codes hinterlegt.

```
#ifndef S7HTIME
#define S7HTIME
typedef struct s7hTime
{
  int is_abs;                 /* 1 = absolute Date and Time
                                 0 = relative time: no Year and month evaluation */
  int sign;                   /* if relative: -1,+1, on absolute: no evaluation */
  int nsec;                   /* Nano seconds  (0..999)  */
  int usec;                   /* Micro seconds (0..999 ) */
  int msec;                    /* Milli seconds ( 0..999) */
  int sec;                    /* Seconds.     [0- .... */
  int min;                    /* Minutes.     [0-59] */
  int hour;                   /* Hours.       [0-23] */
  int mday;                   /* Day.         [1-31] */
  int mon;                    /* Month.       [1-12] */
  int year;                   /* Year             */
  int wday;                   /* Day of week.         [0-6] */
  int yday;                   /* Days in year.[0-365] */
  int isdst;                  /* DST.                 [-1/0/1]*/
  int error;                  /* error code, see list of error codes */
  int error_pos;              /* found Position of the error code */

} S7hTime;

#endif
```

init_s7hTime() Eine Structure s7hTime initialisieren

Erzeuge und initialisiere eine struct 's7hTime'
- Wenn der Pointer *s mit NULL übergeben wird, erzeugt die Funktion
  ein structure und übergibt den Pointer darauf.
  Die structure wird dann initialisiert.
  Der Anwender muss den Speicherplatz der structure dann selbst mit
  free() wieder freigeben

- Wird am Pointer *s eine bereits existierende structure übergeben,
  wird sie nur initialisiert.

struct s7hTime *init_s7hTime (struct s7hTime *s) ;

Hilfsfunktionen zur Zeit-Format-Umrechnung im Rechner

s7hTime_to_tm()

Umwandlung struct 's7hTime' zu struct 'tm'

void s7hTime_to_tm (struct tm *tm, struct s7hTime *s7T) ;

s7hTime_fr_tm()

Umwandlung 'struct tm' zu 'struct s7hTime'
Es wird ein absolutes Datum + Zeit angenommen

void s7hTime_fr_tm (struct s7hTime *s7T, struct tm *tm) ;

s7hTime_to_tv()

Umwandlung struct 's7hTime' zu struct 'timeval tv'

Wenn s7T->is_abs == 1, (absolute Datum und Zeit)
wird s7T->sign nicht ausgewertet.
Wenn s7T->is_abs == 0, (relative Zeit) und s7T->sign == -1
wird tv->tv_sec und tv->usec als negative Zahlen übergeben.
s7T->mon und s7T->year werden ignoriert.

void s7hTime_to_tv (struct timeval *tv, struct s7hTime *s7T) ;

s7hTime_fr_tv()

Umwandlung 'struct timeval' zu 'struct s7hTime'
Mit dem Parameter 'absolut' wird ausgewählt, ob das Zeitformat als absolute Zeit (= 1) oder relative Zeit (= 0) zu interpretieren ist.

retval : 0 = OK
-15 = absolute Zeit, aber negativ

int s7hTime_fr_tv (struct s7hTime *s7T, int absolut, struct timeval *tv) ;

s7hTime_to_ts()

Umwandlung struct 's7hTime' zu struct 'timespec ts'

Wenn s7T->is_abs == 1, (absolute Datum und Zeit)
wird s7T->sign nicht ausgewertet.
Wenn s7T->is_abs == 0, (relative Zeit)
s7T->mon und s7T->year werden ignoriert.

Wenn s7T->sign == -1,
werden tv->tv_sec und tv->usec als negative Zahlen übergeben.

void s7hTime_to_ts (struct timespec *ts, struct s7hTime *s7T) ;

s7hTime_fr_ts()

Umwandlung 'struct timespec' zu 'struct s7hTime'
Mit dem Parameter 'absolut' wird ausgewählt, ob das Zeitformat
als absolute Zeit (= 1) oder relative Zeit (= 0) zu interpretieren ist.

retval : 0 = OK
-10 = absolute Zeit, aber negativ

int s7hTime_fr_ts (struct s7hTime *s7T, int absolut, struct timespec *ts) ;

Relative Zeiten : Uhrzeit-Funktionen

get_s7S5TIME ()

wandelt eine S5TIME (S7: S5T#..) in Millisekunden

uint32_t get_s7S5TIME (void *in) ;

get_s7S5TIME_FMT ()

wandelt eine S5TIME (S7: S5T#..) in Millisekunden
Für nicht benötigte Strukturen muss NULL parametriert werden.
Es wird optional übergeben:
- der Restwert in Nano-Sekunden 0..999999999
- Der Wert in eine structure "timespec *ts"
- Der Wert in eine structure "s7hTime *s7T"

uint32_t get_s7S5TIME_FMT (void *in,
struct timespec *ts, struct s7hTime *s7T);

set_s7S5TIME (*dest, zeit, dimension)

Schreiben eines Zeitwerts in S5T#-Format

Eingabeparameter:
in = Zeitwert in Millisekunden
dim = -1: automatisch berechnen,
sonst 0..3 Zeitbasis ist 10ms,100ms, 1s,10s

Returncode:
0 = OK
-1 = Wert < 0. ausser Bereich
-2 = Wert > 9990sek : ausser Bereich
-3 = Wert mit gewählter Zeitbasis nicht darstellbar

int set_s7S5TIME (void *out,long in, int dim) ;

get_s7TIME()

IEC TIME oder TIME_OF_DAY einlesen
Returnwert in Sekunden

HINWEIS:
Beide Formate beinhalten Zeiten in ms-Einheiten, daher wäre auch der Zugriff direkt mit get_s7DINT() möglich.

Für nicht benötigte Rückgabe-Strukturen muss NULL parametriert werden.

Es wird optional übergeben:
- der Restwert in Nano-Sekunden 0..999999999
- Der Wert in eine structure "timespec *ts"
- Der Wert in eine structure "s7hTime *s7T"

time_t get_s7TIME(void *in, int64_t *ns,
struct timespec *ts, struct s7hTime *s7T) ;

set_s7TIME()

IEC TIME oder TIME_OF_DAY schreiben

HINWEIS:
Beide Formate beinhalten Zeiten in ms-Einheiten, daher wäre auch der Zugriff direkt mit set_s7DINT() möglich.

Eingabe-Werte:
Die Auswahl geschieht durch Übergabe der Eingabe Pointer:
Die Prioritätsreihenfolge ist:
- struct s7hTime *s7T falls NULL dann
- struct timespec *ts falls NULL dann
- time_t time und int64_t ns

void set_s7TIME (void *out, time_t time,int64_t ns,
struct timespec *ts, struct s7hTime *s7T) ;

get_s7LTIME()

S7-1500 LTIME oder LTOD einlesen
Returnwert in Sekunden,
*ns liefert die Nanosekunden
*ts liefert Sekunden und Nanosekunden

HINWEIS:
LTIME und LTOD sind einfach nur INT64-Zahlen, die Nanosekunden enthalten.
Man kann diese auch einfach mit get_LINT() holen und weiterverarbeiten.

Für nicht benötigte Strukturen muss NULL parametriert werden.
Es wird optional übergeben:
- der Restwert in Nano-Sekunden 0..999999999
- Der Wert in eine structure "timespec *ts"
- Der Wert in eine structure "s7hTime *s7T"

time_t get_s7LTIME(void *in, int64_t *ns,

struct timespec *ts,struct s7hTime *s7T) ;

set_s7LTIME()

S7-1500 LTIME oder LTOD ausgeben

'time' in Sekunden, ts übergibt Sekunden und Nanosekunden

HINWEIS:
LTIME und LTOD sind einfach nur INT64-Zahlen, die Nanosekunden enthalten.
Man kann diese auch einfach mit set_LINT() holen und weiterverarbeiten.

Eingabe-Werte:
Die Auswahl geschieht durch Übergabe der Eingabe Pointer:
Die Prioritätsreihenfolge ist:
- struct s7hTime *s7T falls NULL dann
- struct timespec *ts falls NULL dann
- time_t time und int64_t ns

void set_s7LTIME(void *out, time_t time,int64_t ns,
struct timespec *ts, struct s7hTime *s7T) ;

Absolute Zeiten : Datum-und-Uhrzeit-Funktionen

get_s7LDT()

LDT (DATE_AND_LTIME) einlesen: S7-1500

LDT ist effektiv betrachtet eine INT64-Zahl, die die in Nanosekunden gemessene Zeit seit EPOCH (siehe man time(7)) beinhaltet.
Es ist daher eine absolute Zeit. Es sind nur positive Zahlen zulässig.
LDT ähnelt also time_t, beinhaltet aber Nanosekunden statt Sekunden.

Da LDT ein Zeit in ns ist, kann sie innerhalb der S7 auch als mächtige relative Zeit für Zeitrechnungen benutzt werden.

Das folgende Programm ist fast identisch dem von get_s7LTIME(), es wird nur überprüft,
dass t->tv_sec und t->tv_nsec auch positive Zahlen sind, andernfalls , wird als Sekunden (und Nano-Sekunden) -1 ausgegeben.

Returnwert in Sekunden

Für nicht benötigte Strukturen muss NULL parametriert werden.
Es wird optional übergeben:
- der Restwert in Nano-Sekunden 0..999999999
- Der Wert in eine structure "timespec *ts"
- Der Wert in eine structure "s7hTime *s7T"

```
time_t get_s7LDT( void *in, int64_t *ns,
        struct timespec *ts, struct s7hTime *s7T) ;
```

set_s7LDT()

LDT (DATE_AND_LTIME) übertragen: S7-1500

LDT ist effektiv betrachtet eine INT64-Zahl, die die in Nanosekunden gemessene Zeit seit EPOCH (siehe man time(7)) beinhaltet.
Es sind nur positive Zahlen zulässig.
LDT ähnelt also time_t, beinhaltet aber Nanosekunden statt Sekunden.

Es wird von der Zeitzone, der CPU-Zeit und der Implementierung abhängen,
wie diese Zahl wirklich zu interpretieren ist.

Da LDT ein Zeit in ns ist, kann sie innerhalb der S7 auch als mächtige relative Zeit für Zeitrechnungen benutzt werden.

Das folgende Programm ist identisch zu set_s7LTIME(),

Eingabe-Werte:
Die Auswahl geschieht durch Übergabe der Eingabe Pointer:
Die Prioritätsreihenfolge ist:
- struct s7hTime *s7T falls NULL dann

- struct timespec *ts falls NULL dann
- time_t time und int64_t ns

void set_s7LDT(void *out, time_t time,int64_t ns,
struct timespec *ts, struct s7hTime *s7T) ;

get_s7DATE()

IEC DATE einlesen und als Returnwert die Sekunden in EPOCH

Für nicht benötigte Strukturen muss NULL parametriert werden.
Es wird optional übergeben:
- Der Wert in eine structure "timespec *ts"
- Der Wert in eine structure "s7hTime *s7T"

time_t get_s7DATE (void *in,
struct timespec *ts, struct s7hTime *s7T) ;

set_s7DATE()

IEC DATE schreiben
'time' beinhaltet die Sekunden seit EPOCH.

Eingabe-Werte:
Die Auswahl geschieht durch Übergabe der Eingabe Pointer:
Die Prioritätsreihenfolge ist:
- struct s7hTime *s7T falls NULL dann
- struct timespec *ts falls NULL dann
- time_t time. 'time' beinhaltet die Sekunden seit EPOCH.

Falls das Format s7hTime *s7T benutzt wird, bleibt dessen Inhalt erhalten.

void set_s7DATE (void *out, time_t time,
struct timespec *ts, struct s7hTime *s7T) ;

get_s7DT()

Datum-Uhrzeit aus S7 im DATE_AND_TIME-Format (BCD) einlesen
Returnwert in Sekunden
-1 falls ein Fehler erkannt wurde (Tag,Monat < 1)

Das DATE_AND_TIME Format ist in BCD-Bytes
0: Jahr (2stellig) (19)90 .. (20)89
1: Monat (01.12).
2: Tag (01..31)
3: Stunde (00..23)
4: Minute (00.59)
5: Sekunde (00.59)
6 Millisek (2 Höherwertige Stellen 00 .. 99)
7: Millisek , (1 ..7 = sunday .. saturday) (Hi-Nibble,Lo-Nibble)

Für nicht benötigte Strukturen muss NULL parametriert werden.
Es wird optional übergeben:
- der Restwert in Nano-Sekunden 0..999999999
- Der Wert in eine structure "timespec *ts"
- Der Wert in eine structure "s7hTime *s7T"

```
time_t get_s7DT ( void *in, int64_t *ns,
                  struct timespec *ts, struct s7hTime *s7T) ;
```

set_s7DT()

Zeit in S7 schreiben
Simatic S7 DATE_AND_TIME format schreiben.
Schreibt in ein Array, auf das *out zeigt

Das DATE_AND_TIME Format ist
0: Jahr (2stellig) (19)90 .. (20)89
1: Monat (01.12).
2: Tag (01..31)
3: Stunde (00..23)
4: Minute (00.59)
5: Sekunde (00.59)
6 Millisek (2 Höherwertige Stellen 00 .. 99)
7: Millisek , (1 ..7 = sunday .. saturday) (Hi-Nibble,Lo-Nibble)

Eingabe-Werte:
Die Auswahl geschieht durch Übergabe der Eingabe Pointer:
Die Prioritätsreihenfolge ist:
- struct s7hTime *s7T falls NULL dann
- struct timespec *ts falls NULL dann
- time_t time und int64_t ns

```
void set_s7DT ( void *out, time_t time, int64_t ns,
                struct timespec *ts, struct s7hTime *s7T ) ;
```

get_s7DTL()

Datum-Uhrzeit aus S7-1200 / 1500 im DTL-Format einlesen
Returnwert in Sekunden,
-1 falls ein Fehler erkannt wurde (Tag,Monat < 1)

Das DTL Format ist

0:	YEAR	UINT = unsigned int16	[1970 ... = 0]
2:	MONTH	USINT = unsigned char	[1..12]
3:	DAY	USINT = unsigned char	[1..31]
4:	WEEKDAY	USINT = unsigned char	[1..7, 1 = sunday]
5:	HOUR	USINT = unsigned char	[0..23]
6:	MINUTE	USINT = unsigned char	[0..59]
7:	SECOND	USINT = unsigned char	[0..59]
8..11	NANOSECOND	UDINT = unsigned int32	[0..999999999]

Für nicht benötigte Strukturen muss NULL parametriert werden.
Es wird optional übergeben:
- der Restwert in Nano-Sekunden 0..999999999
- Der Wert in eine structure "timespec *ts"
- Der Wert in eine structure "s7hTime *s7T"

```
time_t get_s7DTL ( void *in, int64_t *ns,
                struct timespec *ts, struct s7hTime *s7T  ) ;
```

set_s7DTL()

schreibt eine vorgegebene Zeit im DTL Format zur S7:
in ein Array, auf das *out zeigt

int ns : Wert in Nanosekunden ohne ganze Sekunden

Eingabe-Werte:
Die Auswahl geschieht durch Übergabe der Eingabe Pointer:
Die Prioritätsreihenfolge ist:
- struct s7hTime *s7T falls NULL dann
- struct timespec *ts falls NULL dann
- time_t time und int64_t ns

Das DTL Format ist

0:	YEAR	UINT = unsigned int16	[1970 ...]
2:	MONTH	USINT = unsigned char	[1..12]
3:	DAY	USINT = unsigned char	[1..31]
4:	WEEKDAY	USINT = unsigned char	[1..7, 1 = sunday]
5:	HOUR	USINT = unsigned char	[0..23]
6:	MINUTE	USINT = unsigned char	[0..59]
7:	SECOND	USINT = unsigned char	[0..59]

8..11 NANOSECOND UDINT = unsigned int32 [0..999999999]

Die vorgegebene Zeit muss bei Bedarf vor dem Aufruf dieser Funktion an die Zeitzone und Sommer-Winterzeit angepasst werden.

```
void set_s7DTL (void *out, time_t  time, int64_t  ns,
        struct timespec *ts, struct s7hTime *s7T ) ;
```

2.5 Umwandlungsfunktionen String <-> Zeit (Rechner-Seite)

read_s7TimeStr() : String zu Zeit-Format

Umwandlung eines Eingabestrings in das Format 'struct s7hTime'.

Parameter:
in : Eingabstring
absolut : 1 = absolute Zeit, 0 = relative Zeit,
für relative Zeite sind negative Werte erlaubt.
fmt : Erwartetes Format-String
Darin der Tokenaufbau:
'%<format><delim> wobei <format> :
Y = Jahreszahl
m = Monat
d = Tag
H = Stunde
M = Minute
S = Sekunde
mS = Millisekunde
uS = Mikrosekunde
nS = Nanosekunde
und <delim> ein Zeichen oder Stringende

Beispiele:
"%d.%m.%Y %h:%M:%S.%mS.%uS.%ns" für ein volles Datumsformat
Ausgabe z.B: "03.05.2018 23:54:23.521.476.883"
"%Y-%m-%d" für ein IEC-Datum
Ausgabe z.B: "2018-03-31"
"%-%d %H:%M:%S" für eine Relative Zeit mit Vorzeichen
Ausgabe z.B: "+2 23:45:56"
"%-%dTage %Hh %Mm %Ss" für eine Relative Zeit mit Vorzeichen
Ausgabe z.B: "+56Tage 23h 45m 56s"

Bedingungen für den Format-Aufbau;
- Falls der Formatstring Sekunden und ggf Bruchteile von Sekunden
und auch Stunden oder Minuten beinhaltet, so müssen die Delimiter
von Sekunden und Minuten, Stunden unterschiedlich sein.

- Die Reihenfolge bei Zeiten ist fest: Stunden,[Minuten[,Sekunden[,Milli,[mikro[,Nano]]]]]

Ablauf der Interpretation:
Wenn eine *__absolute Zeit__* erwartet wird, wird von links nach rechts gesucht:
<date> <hour> <minutes> <seconds> <fractions_of_seconds>
<sign> darf im Formatstring nicht enthalten sein,
<datum> ist Pflicht, die Felder für %Y %m %d können in beliebiger
Reihenfolge sein.
<hour> <minutes> <seconds> <fractions_of_seconds> sind optional,
aber wenn eine Einheit existiert, müssen auch die grösseren Einheiten

existieren:
z.B: Wenn %mS dann ist %S notwendig
wenn %S dann ist %M notwendig
wenn %M dann ist %H notwendig.

Die Werte für regulär nicht existierende Felder werden mit 0 besetzt.

Für Datumsangaben:
Die Reihenfolge ist beliebig,
Es dürfen die gleichen delimiter wie bei Sekunden benutzt werden, andere sind natürlich auch zulässig.
Ein Datum wird nur ausgewertet, wenn alle 3 Teile Jahr,Monat,Tag sowohl im Formatstring als auch in der String-Eingabe vorhanden sind.

Wenn eine *__relative Zeit__* erwartet wird, dürfen Jahre und Monate nicht enthalten sein. Tag ist erlaubt.

<sign> <day> <hour> <minutes> <seconds> <fractions_of_seconds>
Die Funktion sucht als erstes nach den Sekunden.
Falls gefunden, werden die Sekundenbruchteile gesucht.
Danach sucht das Programm Richtung links nach Minuten, Stunden, Tage.

Falls der Eingabestring nur eine Zahl beinhaltet, so wird sie als Sekunden interpretiert.

Die Werte für regulär nicht existierende Felder werden intern mit 0 besetzt.

Ausgabeformat:
Die Structure 'struct s7hTime *s7T' wird gefüllt.

Retval = 0 : OK,
sonst <0 = Formatfehler
genauere Auswertung in s7T->error
s7T->error_pos

```
int read_s7TimeStr (const char * in, const int absolut, const char *fmt,
        struct s7hTime *s7T) ;
```

Parameter:

const char * in	= Eingabestring
const int absolut	= 0 : relative Zeit erwartet, 1: absolute Zeit erwartet
const char *fmt	= erwartetes Format
struct s7hTime *s7T	= Ausgabe-Structure

strfs7Time() : Zeit-Format zu String

Formatierte Ausgabe von Datum und Zeit. absolute und relative Zeit.

Die Funktion orientiert sich an strftime(), kennt zwar nur eine Untermenge der Formatanweisungen, stellt aber zusätzliche zur Verfügung.

Als Eingangsdaten übernimmt strfs7Time nur eine structure vom typ s7hTime.

Erlaubte Formatzeichen:
- falls erstes Zeichen in Fomatstring, dann wird das Vorzeichen ausgegeben.

%+ ,%- '+' Vorzeichen wird immer dargestellt, '-' Vorzeichen wird bei negativer Zeit dargestellt
%y : Jahr ohne Jahrhundert (2 Zeichen)
%Y : Jahr (4 Zeichen)
%m : Monat
%d : Tag
%H : Stunde
%M : Minute
%S : Sekunde
%mS : 0..999 ms
%uS : 0..999 us
%nS : 0..999 ns

Returncode:
dstr, gefüllt mit formatierter Ausgabe.

Fehler:
Die Fehlernummer wird in struct s7hTime s7T-error zurückgeschrieben,
sie kann hinter dem Aufruf dieser Funktion ausgewertet werden.
0 = kein Fehler.
< 0 : Siehe char * s7hTime_errTxt (int error)

```
char* strfs7Time(char dstr[], size_t dstr_len, const char fmt[], \
    struct s7hTime *s7T) ;
```

Parameter:
const char dstr[] = Ausgabestring
size_t dstr_len = Maximale Länge des Ausgabestrings
const char *fmt = erwartetes Format
struct s7hTime *s7T = Ausgabe-Structure

2.6 Hilfsfunktionen

Fehlernummern der Funktionen als Text ausgeben

s7hTime_errTxt() : Fehlernummer als Text deutsche Version

generiert aus Fehlernummern einen Fehlertext
verwendbar hinter
read_s7TimeStr ()
s7hTime_fr_tv ()
etc.

error codes :

-1 : set_s7S5TIME () : Wert < 0. ausser Bereich
-2 : set_s7S5TIME () : Wert > 9990sek : ausser Bereich
-3 : set_s7S5TIME () : Wert mit gewählter Zeitbasis nicht darstellbar
-4 : set_s7COUNTER () : Wert < 0. ausser Bereich
-5 : set_s7COUNTER () : Wert > 999 : ausser Bereich
-6 : set_s7REAL () : Wert > 3.402823E+38 : ausser Bereich
-7 : set_s7REAL () : Wert < 1.175495E-38 : ausser Bereich
-8 : set_s7LREAL () : Wert > 1.7976931348623158e+308 : ausser Bereich
-9 : set_s7LREAL () : Wert < 2.2250738585072014e-308 : ausser Bereich
-10 : get_s7STRING () : Längenangaben in S7 nicht plausibel <length> Bytes gelesen
-11 : get_s7STRING () : Zielstring zu kurz, nur dmax -1 Bytes gelesen
-12 : set_s7STRING () : Fehler, ungültige Länge smax
-13 : set_s7STRING () : Quelle zu lang für Zielstring,wurde gekürzt übertragen

-15 : s7hTime_fr_tv () : Absolute Zeit aber negativen Wert gelesen

-20 : read_s7TimeStr () : Absolute Zeit (Datum) aber Jahr-Feld fehlt im Format
-21 : read_s7TimeStr () : Absolute Zeit (Datum) aber Monat-Feld fehlt im Format
-22 : read_s7TimeStr () : Absolute Zeit (Datum) aber Tag-Feld fehlt im Format
-23 : read_s7TimeStr () : Absolute Zeit (Datum) aber Vorzeichenfeld im Format
-24 : read_s7TimeStr () : Absolute Zeit Minuten definiert aber Stunden fehlt im Format
-25 : read_s7TimeStr () : Absolute Zeit Sekunden definiert aber Minuten fehlt im Format
-26 : read_s7TimeStr () : Absolute Zeit ms definiert aber Sekunden fehlt im Format
-27 : read_s7TimeStr () : Absolute Zeit us definiert aber Millisekunden fehlt im Format
-28 : read_s7TimeStr () : Absolute Zeit ns definiert aber Mikrosekunden fehlt im Format
-29 : read_s7TimeStr () : Absolute Zeit Jahr fehlt in Eingabestring
-30 : read_s7TimeStr () : Absolute Zeit Monat fehlt in Eingabestring
-31 : read_s7TimeStr () : Absolute Zeit Tag fehlt in Eingabestring
-32 : read_s7TimeStr () : Relative Zeit aber Jahr-Feld im Format
-33 : read_s7TimeStr () : Relative Zeit aber Monat-Feld im Format
-34 : read_s7TimeStr () : Datum-Eingabe erwartet, aber in Eingabestring nicht enthalten.

-40 : strfs7Time () : Absolute Zeit aber kein vollständiges Datums-Format.
-41 : strfs7Time () : Relative Zeit aber format beinhaltet Datums-Format.
-42 : strfs7Time () : Vorzeichen in Formatstring nicht auf erster Position.
-43 : strfs7Time () : Unvollständiges Sekundenformat.

-44 : strfs7Time () : Reihenfolge der Zeit-Tokens nicht absteigend.

char * s7hTime_errTxt (int error) ;

s7hTime_errTxt_En() : Fehlernummer als Text ausgeben Englische Version

For the time functions : English version
generates an error text from an error number
usable behind
read_s7TimeStr ()
s7hTime_fr_tv ()
etc.

error codes :

-1 : set_s7S5TIME () : value < 0. violates limits
-2 : set_s7S5TIME () : value > 9990 sec : violates limits
-3 : set_s7S5TIME () : value nor processible with the selected time base (0..3)
-4 : set_s7COUNTER () : value < 0. violates limits
-5 : set_s7COUNTER () : value > 999 : violates limits
-6 : set_s7REAL () : value > 3.402823E+38 : violates limits
-7 : set_s7REAL () : value < 1.175495E-38 : violates limits
-8 : set_s7LREAL () : value > 1.7976931348623158e+308 : violates limits
-9 : set_s7LREAL () : value < 2.2250738585072014e-308 : violates limits
-10 : get_s7STRING () :length informations in S7 not plausible, <length> bytes read
-11 : get_s7STRING () : dest[] is too short.only dmax -1 bytes read
-12 : set_s7STRING () : error: invalid length smax
-13 : set_s7STRING () : src[] too long for destination, was truncated

-15 : s7hTime_fr_tv () : absolute date and time aber negativen value gelesen

-20 : read_s7TimeStr () : absolute date'n time but absence of year field in format
-21 : read_s7TimeStr () : absolute date'n time but absence of month field in format
-22 : read_s7TimeStr () : absolute date'n time but absence of day field in format
-23 : read_s7TimeStr () : absolute date'n time but sigh field in format
-24 : read_s7TimeStr () : absolute date'n time minutes defined but hours are absent in format
-25 : read_s7TimeStr () : absolute date'n time seconds defined but minutes are absent in format
-26 : read_s7TimeStr () : absolute date'n time ms defined but seconds are absent in format
-27 : read_s7TimeStr () : absolute date'n time us defined but ms are absent in format
-28 : read_s7TimeStr () : absolute date'n time ns defined but us are absent in format
-29 : read_s7TimeStr () : absolute date'n time year is absent in input string
-30 : read_s7TimeStr () : absolute date'n time month is absent in input string
-31 : read_s7TimeStr () : absolute date'n time day is absent in input string
-32 : read_s7TimeStr () : relative time but year field in format
-33 : read_s7TimeStr () : relative time but month field in format
-34 : read_s7TimeStr () : estimated date input but not provided by input string

-40 : strfs7Time () : absolute date'n time but format does not contain a complete date definition
-41 : strfs7Time () : relative time but format contains date format

-42 : strfs7Time () : sing in format string but not at first position
-43 : strfs7Time () : unclomplete format of seconds
-44 : strfs7Time () : no down going sequence of time fields in format

char * s7hTime_errTxt_En (int error) ;

Für Programmtest : print_s7hTime (struct s7hTime *p)

Die Funktion read_s7TimeStr () sollte sowohl flexibel als auch universell sein, das ermöglicht natürlich eine Menge von Fehlinterpretationen.
print_s7hTime () soll dabei helfen, read_s7TimeStr () richtig anzuwenden.

3. s5types.c, s5types.h alte FUNKTIONEN

```
/* ===========================================================================

   Funktionen Datenaustausch von Simatic S5/S7-DBs mit Unix/Linux

   functions for data exchange with a Simatic S5/S7 and Linux

* ===========================================================================
   (c) Heisch Automatisierungstechnik, Werner Heisch

       http://sites.inka.de/heisch
       http://www.heisch-automation.de

 !!! Das Array, das fuer den Transfer von oder zu der S5 vorgesehen ist,
      sollte als unsigned short definiert werden.
      Dadurch bleibt der Zusammenhang Array<-> DB erhalten, was die
      Fehlersuche entscheidend erleichtert.

      Die folgenden functions funktionieren aber auch mit unsigned char,
      diese Darstellung wird für die Kommunikation mit der S7 empfohlen.

      ------------

      The array, which is used for data transfer with Simatic S5, should be
      defined as an array of unsigned short. This helps to find errors.

      The functions also work with arrays of unsigned char, which is
      recommended for a link to Simatic S7.


*/
```

3.1 Integer and BCD Formate

```
/* ============  S5types_i.c /.h  ========================================

   Funktionen Datenaustausch von Simatic S5-DBs mit Unix/Linux
   functions for data transfer beween Simatic S5 / S7 and Linux
         -------------- Integer-part -----

   S5:  KH,KF,KD,KT,KZ -----
   S7:  WORD,INT,DWORD,DINT,S5TIME,C

* ========================================================================

*/


#ifndef MATH_H
  #include <math.h>
#endif

/* --------------- get_dd () ------------------------- */
/* Doppelwort als unsigned long einlesen
   importiert eine unsigned Zahl
   ---
   to get a double word as unsigned number

*/

unsigned long get_dd(void *in)
;
/* --------------- get_dd_kf () ------------------------- */
/* Doppelwort als long integer einlesen
   exportiert eine signed-Zahl
   ---
   to get a double word as signed long
*/

long get_dd_kf(void *in)
;
/* --------------- get_dw () ------------------------- */
/*
 importiert eine unsigned-Zahl
 ---
 to get a word (16bit) as a unsigned int.
*/
int get_dw (void *in)
;
/* --------------- get_dw_kf () ------------------------- */
/*
 importiert eine signed-Zahl
 ---
 to get a word (16Bit) as a signed int
 */
int get_dw_kf (void *in)
;
/* --------------- get_kt () ------------------------- */
```

```
/*
importiert einen Zeitwert (S5: KT,S7: S5T#..) in Millisekunden
---
to get a time value ( S5: KT, S7: S5t# ) im milliseconds

*/
long get_kt (void *in)
;
/* --------------- get_kz () ------------------------- */
/*
 importiert einen Zaehlerwert
 ---
 to get a counter value
*/
int get_kz (void *in)
;
/* --------------- get_dl () ------------------------- */
/*
S5: importiert den linken Teil eines DW (Simatic S5 only)
---
S5: to get the left part of a DW
*/
int get_dl (void *in)
;
/* --------------- get_dr () ------------------------- */
/*
S5: importiert den rechten Teil eines DW  (Simatic S5 only)
---
S5: to get the right part of a DW
*/
int get_dr (void *in)
;
/* *************** Transfers in Richtung SPS ******************** */
/*                 transfers in direction to plc                  */

/* --------------- set_dw (*ziel,quelle) ------------------------- */
/* Schreiben eines 16-Bit-KF-Werts in ein DW / DBW
   ---
   writes a signed or unsigned short to a DW / DBW
*/

int set_dw (void *out,int in)
;
/* --------------- set_dd (*ziel,quelle) ------------------------- */
/* Schreiben eines 32-Bit-KF-Werts zu einem DD / DBD
   ---
   writes a 32bit value to a DD / DBD
*/
int set_dd (void *out,int in)
;
/* --------------- set_kz (*ziel, quelle) ------------------------ */
/* Schreiben eines Zaehlerwerts
 bei Wert-Ueberlauf: Rueckgabe = -1; sonst = 0;
 ---
 writes to a counter value
 on overflow: returncode = -1 else return 0
*/
```

```
int set_kz (void *out,int in)
;
/* --------------- kt_to_short (*ziel, quelle, dimension) --------------------
*/
/* Schreiben eines Zeitwerts in BCD-codierte short-Zahl
   ( Bytes sind noch nicht gedreht, set_dw anschliessend erforderlich)
   Eingabeparameter:
     in  = Zeitwert in Millisekunden
     dim = -1: automatisch berechnen,
                 sonst 0..3 Zeitbasis ist 10ms,100ms,s,10s
   bei Wert-Ueberlauf oder dim-Fehler: Rueckgabe = -1; sonst zeitwert in
   KT-Format / S5t#- Format;
   ---
   writes a time value  to a BSC coded short
   ( bytes are not rotated, set_dw is necessary, after this fuction )
   Input parameters:
     in =  time in milli seconds
     dim = -1: automatic calculation,
               else 0..3 timebase is 10ms,100ms,1s,10s
   on value overflow or dim error : return code = -1 else the value
   in KT format / S5T# format is returned
*/

short kt_to_short (long in, int dim)
;
/* --------------- set_kt (*ziel, quelle, dimension) -------------------- */
/* Schreiben eines Zeitwerts in KT / S5t#-Format
   Eingabeparameter:
     in  = Zeitwert in Millisekunden
     dim = -1: automatisch berechnen,
                 sonst 0..3 Zeitbasis ist 10ms,100ms,s,10s

   bei Wert-Ueberlauf: Rueckgabe = -1; sonst = 0;
   ---
   to write a time value in KT / S5t#-format
   input parameters:
     in  =  time in milli seconds
     dim = -1: automatic calculation,
               else 0..3 timebase is 10ms,100ms,1s,10s
   on overflow: returncode = -1; else = 0;

*/

int set_kt (void *out,long in, int dim)
;
```

3.2 Gleitpunkt-Formate

```
/* ============== S5types_fp.c / .h   ====================================

   Funktionen Datenaustausch von Simatic S5/S7-DBs mit Unix/Linux
   functions for data transfer beween Simatic S5 / S7 and Linux
    -------------- Floating point part -----

   S5:  KG  (Simatic S5 floating point)
   S7:  REAL


   Version 04.01.2002 (Comments) code: 02.08.2000 HW
* ========================================================================
*/


#ifndef _UNISTD_H
  #include <unistd.h>
#endif
#ifndef MATH_H
  #include <math.h>
#endif

/* --------------- get_real() ------------------------- */
/*
 S7-REAL-Zahl als double einlesen
 ---
 to get a S7-real in double format
*/
/* returncode : 0 = ohne Fehler, -1 = fehler */
/* returncode : 0 = no fault, -1 = with fault */

int get_real(double *retval,void *in)
;
/* --------------- get_s5kg() ------------------------- */
/*
 S5-REAL-Zahl als double einlesen
 ---
 to get a S5-real as double format
*/
/* returncode : 0 = ohne Fehler, -1 = fehler */
/* returncode : 0 = no fault, -1 = with fault */

int get_s5kg(double *retval,void *in)
;
/* *************** Transfers in Richtung SPS ******************** */
/*                 transfers in direction to plc                  */

/* --------------- set_real (*ziel,quelle) ------------------------ */
/* Schreiben einer double zu einer S7-REAL-Zahl
   ---
   write a double to a S7-real

returncode : 0 = OK; -1 = OUT OF RANGE
*/
```

```
int set_real (void *out, double val)
;
/* --------------- set_s5kg (*ziel,quelle) ------------------------- */
/* Schreiben einer double zu einer S5-KG-Zahl
   ---
   write a double to a S5-KG-value

returncode : 0 = OK; -1 = OUT OF RANGE

*/

int set_s5kg (void *out, double val)
;
```

3.3 Datum-Uhrzeit-Formate

```
/* ============== S5types_t.c / .h   ====================================

   Funktionen Datenaustausch von Simatic S5/S7-DBs mit Unix/Linux
   functions for data transfer beween Simatic S5 / S7 and Linux
    -------------- DATE and TIME -Functions -----

   S5:  RealTimeClock-Format of S5-095U, S5-115U,  -----
   S7:  (IEC)DATE, (IEC)TIME , DATE_AND_TIME(BCD)

   Stand / Version
   02.08.2002 HW DATE_AND_TIME eingebaut
   02.01.2002 HW
   01.01.2002 HW
   22.06.2001 HW
   19.06.2001 HW
   27.02.2001 HW
   07.05.2000 HW

* ========================================================================
  (c) Heisch Automatisierungstechnik, Werner Heisch
       http://sites.inka.de/heisch
       http://www.heisch-automation.de



   PROBLEM: die Simaticen laufen normalerweise in Ortszeit,
  ( sie werden von Step 7(TM) aus gestellt ), stellen aber selbständig
   keine Sommer-Winterzeit um.
   (Der Algorithmus ist schliesslich keine Naturkonstante sondern
   abhaengig von politischen Vorgaben. )
   Es ist folglich nicht zu erkennen, mit welcher Uhrzeit
   (UTC, lokale Zeit,lokale Sommerzeit) die Simatic wirklich laeuft.
   Das muss bei Bedarf der Anwender dieser function selbst entscheiden
   und ggf. korrigierend eingreifen.
   Die unten stehende function liefert den Zeitwert so zurueck,
   wie er aus der Simatic gelesen wurde. Dies wird in der Regel die aktuelle
   lokale Uhrzeit sein.

   Die rückgelieferten Zeitwerte  time_t get_iec_dateandtime() oder das
   structure-Element tv->tv_sec beinhalten beide jeweils die Zeit in Sekunden
   seit EPOCH, aber (vermutlich) als lokale (Sommer ?? )- Zeit.
   Die Rueckgabewerte sind also in einem Format(time_t, struct timeval) ,
   in dem normalerweise der Zeitbezug EPOCH und UCT ist.

   Die C-functions, die timeval tv weiterverarbeiten ( z.B. gmtime(),
   localtime() erwarten eine Zeit auf UTC basierend.

   Wenn davon ausgegangen wird, dass die Simatic-Uhr in der lokalen
   nicht-Sommerzeit laeuft, dann kann die Zeit einfach auf UTC korrigiert
   werden:
   Fuer Laender oestlich von London(Greenwich) : Stunden abziehen
   Fuer Laender westlich von London(Greenwich) : Stunden dazaehlen

   Beispiel: Simatic S7 steht in Düsseldorf -> Deutschland -> 1 Stunde östlich
     (time_t) uct_time =  get_iec_dateandtime( .. ) - 3600;
```

```
Beispiel: Simatic S7 steht in Havanna -> Kuba -> 5 Stunden westlich
  (time_t) uct_time =  get_iec_dateandtime( .. ) + 5 * 3600;

Wenn aber zum Beispiel eine Zeitsynchronisation von einem Leitsystem aus
erfolgt, wird's schwieriger, es muß dann bekannt sein ob die Simatic-Uhr
entsprechend Sommer-/Winterzeit umgestellt wird und entsprechend
zurückkorrigiert werden muß.

Die Routinen zum Synchronisieren der Zeit der Simatic sind hinsichtlich
Zeitzone und Sommer-Zeit-Umschaltung flexibel. Dies wird über den
Parameter local gesteuert.
local = 0  :  Simatic wird mit UTC synchronisiert
      = 1  :  Simatic wird mit lokaler Zeit ohne Beruecksichtigung der
              lokalen Sommerzeit synchronisiert
      = 2  :  Simatic wird mit lokaler Zeit unter Beruecksichtigung der
              lokalen Sommerzeit synchronisiert
      sonst: <zahl> != 0,1  Korrekturfaktor zu UTC: ohne Beruecksichtigung
              der lokalen Sommerzeit
              Beispiel: Simatic in Duesseldorf soll OHNE
              Sommerzeitumschaltung betrieben werden:
              Duesseldorf ist eine Stunde vor Greenwich -> local = 3600


----

PROBLEM: The simatic processors normaly run in local time ( the are set by
Step 7(TM) but do not change automaticly to daylight saving time.
( The algorithm depends from political decisions )
Therfore it is impossible to know, in which time the the Simatic runs really
and it is impossible to correct the time automaticly.
The function below returns the time, as it is read from the Simatic.
Normally it is the local time ... ?
The returnvalues are in a format (time_t, struct timeval) which normal
context is EPOCH and UTC.
The functions, for example, wich handle timeval-structures, expect
times based on UTC.

If the Simatic runs in local time but not in  daylight saving time the
correction is rather easy to do:
For countries in the east of London (greenwich) subtract hours.
For countries in the west of London (greenwich) subtract hours.

example: Simatic S7 is in Duesseldorf -> Germany -> 1 hour in the east
  (time_t) uct_time =  get_iec_dateandtime( .. ) - 3600;

( If I use the same example as above, George W. will kill me :-)
example: Simatic S7 is situated in Miami -> USA -> 5 hours in the west
  (time_t) uct_time =  get_iec_dateandtime( .. ) + 5 * 3600;

But: wenn the time is synchronized bei a process control system, it is more
difficult: It has to be known, if the time is changed according to
daylight saving time an eventually a correction is needed.


The functions for synchronizing the simatic are flexible, they are
controled by the parameter "local".
local = 0  :  Simatic will be synchronized with UTC
```

```
         = 1  :  Simatic will be synchronized with local time without
                 regarding the daylight saving time
         = 2  :  Simatic will be synchronized with local time regarding
                 the daylight saving time
         else:  <zahl> corretion value according to UTC: without regarding
                 any Daylight saving time
                 i.e.: Simatic is in Duesseldorf (Germany) and shall not
                 regard the daylight saving time:
                 Duesseldorf is one hour before Greenwich -> local = 3600


*/

#ifndef _SYS_TIME_H
  #include <sys/time.h>
#endif
#ifndef _SYS_TIMEB_H
  #include <sys/timeb.h>
#endif
#ifndef _UNISTD_H
  #include <unistd.h>
#endif
#ifndef _STDLIB_H
  #include <stdlib.h>
#endif




/* --------------- get_iec_dateandtime() ------------------------- */
/* IEC DATE und IEC TIME einlesen
   (in Simatic: 2 Variablen: IEC_DATE und IEC_TIME)
   export als  returnwert die Sekunden in EPOCH (Calendar)
   und ueber time_val *tv, die genaue Zeit Sekunden,Millisekunden *1000

   Parameter
   local : 0 : Simatic läuft in UTC  Coordinated Universal Time ( = GMT )
           1 : Simatic läuft in local time immer ohne Sommerzeit
           2 : Simatic läuft in local time mit Berücksichtigung der
               Sommerzeit

         sonst : direkter Offset ( Sek) zu UTC ohne Berücksichtigung
             der Sommerzeit

    BEMERKUNG: für Fall 2 kann für die letzte Stunde Sommerzeit natürlich
               nicht entschieden werden, ob es die letzte Stunde Somerzeit
               oder bereits die erste Stunde Winterzeit ist !

       /usr/include/sys/time.h:

       struct timeval {
               long tv_sec;        /* seconds
               long tv_usec;  /* microseconds
       };
```

```
   ----

   Read Simatic-S7 IEC DATE and IEC-TIME
   (in Simatic: 2 variables: IEC_DATE und IEC_TIME)
   and return the value in seconds sice EPOCH. (Calendar)
   returnvalue : seconds, better use the structure timeval *tv, it contains
   the time in seconds and milliseconds *1000

   Parameter
   local : 0 : Simatic runs in UTC  Coordinated Universal Time ( = GMT )
           1 : Simatic runs in local time ignoring daylight saving time
           2 : Simatic runs in local time regarding daylight saving time
           else :
           direct Offset ( sec) to UTC without regarding daylight saving time

   NOTE: in case of 2 ( local time including DST) we can not decide
         correctly for the last hour in DST-phase !!

       /usr/include/sys/time.h:

       struct timeval {
               long tv_sec;        /* seconds
               long tv_usec;  /* microseconds
       };


*/

time_t get_iec_dateandtime( void *in, struct timeval *tv,int local)
;




/* --------------- get_iec_date() ------------------------- */
/* IEC DATE einlesen und
   export als  returnwert die Sekunden in EPOCH

   Da nur das Datum übermittelt wird und deshalb keine Informationen über die
   Uhrzeit vorliegen, ist eine Zeitzonen-Korrektur nicht möglich.
   Das Datum wird weitergegeben wie es ist.

   ----

   Read Simatic-S7 IEC DATE and return the value in seconds sice EPOCH.
   returnvalue : seconds since EPOCH

   because there is no time-of-day information it is impossible to
   execute a timezone correction. Therfore this date will not be changed.

*/

time_t get_iec_date( void *in, struct timeval *tv)
;
```

```
/* --------------- get_iec_time() ------------------------- */
/* IEC TiME einlesen
   returnwert in Sekunden, tv liefert Sekunden und Mikrosekunden

       /usr/include/sys/time.h:

       struct timeval {
               long tv_sec;        /* seconds
               long tv_usec;  /* microseconds
       };


   ----

   Read Simatic-S7 IEC TIME and return the value in seconds,
   tv returns seconds und useconds

       /usr/include/sys/time.h:

       struct timeval {
               long tv_sec;        /* seconds
               long tv_usec;  /* microseconds
       };


*/

unsigned long get_iec_time( void *in, struct timeval *tv)
;




/* --------------- get_s7_dateandtime() ------------------------- */
/* Datum-Uhrzeit aus S7 im DATE_AND_TIME-Format (BCD) einlesen
   returnwert in Sekunden, tv liefert Sekunden und Mikrosekunden  (UCT)
              -1 falls ein Fehler erkannt wurde (Tag,Monat < 1)

   Parameter
   local : 0 : Simatic läuft in UTC  Coordinated Universal Time ( = GMT )
           1 : Simatic läuft in local time immer ohne Sommerzeit
           2 : Simatic läuft in local time mit Berücksichtigung der
               Sommerzeit

           sonst :
            direkter Offset ( Sek) zu UTC ohne Berücksichtigung der Sommerzeit

    BEMERKUNG: für Fall 2 kann für die letzte Stunde Sommerzeit natürlich
               nicht entschieden werden, ob es die letzte Stunde Sommerzeit oder
               bereits die erste Stunde Winterzeit ist !

   Das DATE_AND_TIME Format ist
    0:  Jahr (2stellig) (19)90 .. (20)89
    1:  Monat  (01.12).
```

```
 2:  Tag    (01..31)
 3:  Stunde (00..23)
 4:  Minute (00.59)
 5:  Sekunde (00.59)
 6   Millisek (2 Höherwertige Stellen 00 .. 99)
 7:  Millisek , ( 1 ..7 = sunday .. saturday )    (Hi-Nibble,Lo-Nibble)

    /usr/include/sys/time.h:

    struct timeval {
            long tv_sec;    /*    seconds
            long tv_usec;  /* microseconds
    };

    struct timeb {
            time_t time;
            unsigned short millitm;
            short timezone;
           short dstflag;
    }; <<<<<<<<<<<< see man ftime(3)


----

Read Simatic-S7 in DATE_AND_TIME-Format (BCD)and return the value in seconds,
tv returns seconds und useconds (UCT)
         -1 if an error is detected ( day or month < 1)

Parameter
local : 0 : Simatic runs in UTC  Coordinated Universal Time ( = GMT )
        1 : Simatic runs in local time ignoring daylight saving time (DST)
        2 : Simatic runs in local time regarding daylight saving time (DST)
        else :
         direct Offset ( sec) to UTC without regarding daylight saving time

NOTE: in case of 2 ( local time including DST) we can not decide correctly
      for the last hour in DST-phase !!


The DATE_AND_TIME format is
 0:  year
 1:  Month
 2:  day
 3:  hour
 4:  minute
 5:  second
 6   Millisek *10
 7:  Millisek , ( 1 ..7 = sunday .. saturday )    (Hi-Nibble,Lo-Nibble)

    /usr/include/sys/time.h:

    struct timeval {
            long tv_sec;        /* seconds
            long tv_usec;  /* microseconds
    };

    struct timeb {
```

```
               time_t time;
               unsigned short millitm;
               short timezone;
              short dstflag;
       }; <<<<<<<<<<< see man ftime(3)


*/

time_t get_s7_dateandtime ( void *in, struct timeval *tv,int local)
;
```

```
/* --------------- set_s7_dateandtime() ------------------------- */
/* Simatic S7 DATE_AND_TIME format schreiben
   schreibt in ein Array, auf das *out zeigt

   Das DATE_AND_TIME Format ist
    0:  Jahr (2stellig) (19)90 .. (20)89
    1:  Monat  (01.12).
    2:  Tag    (01..31)
    3:  Stunde (00..23)
    4:  Minute (00.59)
    5:  Sekunde (00.59)
    6   Millisek (2 Höherwertige Stellen 00 .. 99)
    7:  Millisek , ( 1 ..7 = sunday .. saturday )    (Hi-Nibble,Lo-Nibble)

   local : 0 : Resultierende Zeit ist UTC  Coordinated Universal Time ( = GMT )
           1 : Resultierende Zeit ist local time mit Berücksichtigung der
               Sommerzeit
           sonst :
             direkter Offset ( Sek) zu UTC ohne Berücksichtigung der Sommerzeit

       /usr/include/sys/time.h:

       struct timeval {
               long tv_sec;        /* seconds
               long tv_usec;  /* microseconds
       };


   ----

   Write to Simatic-S7  in DATE_AND_TIME format (BCD)
   Writes to an array of 8 bytes, beginning with the byte pointed to.

   The DATE_AND_TIME format is
    0:  year
    1:  Month
    2:  day
    3:  hour
    4:  minute
    5:  second
    6   Millisek *10
    7:  Millisek , ( 1 ..7 = sunday .. saturday )    (Hi-Nibble,Lo-Nibble)


   local : 0 : resulting time is UTC  Coordinated Universal Time ( = GMT )
           2 : resulting time is local time regarding daylight saving time
           else :
             direct Offset ( sec) to UTC without regarding daylight saving time

   return 0, if OK


       /usr/include/sys/time.h:

       struct timeval {
               long tv_sec;        /* seconds
               long tv_usec;  /* microseconds
```

```
        };

*/

int set_s7_dateandtime (void *out,int local)
;
```

```
/* --------------- get_95u_time() ------------------------- */
/* Zeit aus S5-95U einlesen
   returnwert in Sekunden

        ---     Wochentag ( 1 ..7 = Sonntag .. Samstag )
       tag,     Monat
       jahr,    Stunde ( Bit 7 = AM/PM
       Minute,  Sekunde
   ----

   Read Simatic-S5 95U  TIME and return the value in seconds,
   dow returns day of the week

   The time is read
    0:  ---      day_of_week ( 1 ..7 = sunday .. saturday )
    1:  day,     month
    2:  year     hour   ( bit 7 is AM/PM)
    3:  minute   second

*/

time_t get_95U_time( void *in, int *dow)
;



/* --------------- sync_95u_time() ------------------------- */
/* Zeit in S5-95U / 115U synchronisieren
   Diese function ist zum Synchronisieren der Uhr in
   - S5-95U
   - S5 115 U ( CPU943 und 944 mit 2 seriellen Schnittstellen )
   - S5 115 U  CPU945

   schreibt in ein Array, auf das *out zeigt
   ampm == 0 : Stunden 0..23
        != 0 : Stunden 1..12 AM, 1..12 PM
   local : 0 : Resultierende Zeit ist UTC  Coordinated Universal Time ( = GMT )
           1 : Resultierende Zeit ist local time mit Berücksichtigung der
               Sommerzeit
           sonst :
             direkter Offset ( Sek) zu UTC ohne Berücksichtigung der Sommerzeit

   return 0, falls OK

   Das Zeit-Array für die Simatic
        ---     Wochentag ( 1 ..7 = Sonntag .. Samstag )
       tag,     Monat
       jahr,    Stunde ( mit 7 = AM/PM
       Minute,  Sekunde

   ----

   synchronize Simatic-S5 95U / 115U TIME

   This function synchronizes the clock in
   - S5-95U
   - S5 115 U ( CPU943 und 944 with 2 serial ports )
```

```
   - S5 115 U  CPU945

   writes to an array to which points *out
   ampm == 0 : hours 0..23
        != 0 : hours 1..12 AM, 1..12 PM

   local : 0 : resulting time is UTC  Coordinated Universal Time ( = GMT )
           1 : resulting time is local time regarding daylight saving time
           else :
           direct Offset ( sec) to UTC without regarding daylight saving time

   return 0, if OK

   The time array for Simatic
    0:  ---      day_of_week ( 1 ..7 = sunday .. saturday )
    1:  day      month
    2:  year     hour   ( bit 7 is AM/PM )
    3:  minute   second

*/

int sync_95U_time (void *out, int ampm,int local)
;
```

3.4. String-Formate

```
/* ============  S5types_c.c /.h  ========================================

   Funktionen Datenaustausch von Simatic S5-DBs mit Unix/Linux
   functions for data transfer beween Simatic S5 / S7 and Linux
         ------------- Character-part -----

   S5:  KC -----
   S7:  STRING

* ============================================================================

*/


/* --------------- get_S7string () ------------------------- */
/*  das Simatic-S7-STRING "*in" in ein string dest[] einlesen,
    maximale Länge des Zielstrings = dmax;

    return code : >= 0 : Anzahl der gelesenene Zeichen in dstr[]
                : == -1 ; Fehler

   ---
   to get the Simatic S7 string "*in" into the string dest[],
   maximum length of the destination string is dmax.

   return code : >= 0 :count of read characters in dstr[]
                : == -1 ; Fehler

*/

int get_S7string(void *in,char dstr[],int dmax)
;

/* --------------- set_S7string () ------------------------- */
/*  das String src[] in das Simatic-S7-STRING "*out" schreiben,
    maximale Länge des Zielstrings = smax;
    smax muß mit der Datendeklaration im S7-DB übereinstimmen,
    z.B: in DB: STRING[5]  -> smax = 5.

    return code : == 0  : kein Fehler
                : == -1 : Fehler, ungültige Länge smax:
                          gültige smax = 1.. 254
                  == +1 : Warnung: Quelle zu lang für Zielstring,
                          wurde gekürzt übertragen.

   ---

   write the string src[] into the Simatic S7 string "*in",
   maximum length of the destination string is smax.
   smax has to be indentical with the data declaration of the Simatic
   datablock,
   i.e. : in DB: STRING[5]   -> smax = 5.

   return code : ==  0  : no error
```

```
                : == -1  : error, invalid length smax
                         : valid is smax = 1..254
                : == +1  : warning: source string to long for destination
                           truncated while transferring

*/

int set_S7string(void *out,const char src[],int smax)
;
```

3.5 Hilfsfunktionen Strings

```
/* ******************************************************

   String-funktionen als Hilfsfunktionen für convert_lib

   String functions for convert_lib

Stand: 16/11/1998 HW ff
       19.02.2001 HW
       10.09.2002 HW
       18.06.2007 HW Anpassung an Suse 10.1  gcc 4.1.0
       02.04.2016 HW Erweiterung um %uS

 ****************************************************** */


#ifndef _STRING_H
  #include <string.h>
#endif

#ifndef _CTYPE_H
  #include <ctype.h>
#endif

#ifndef _STDLIB_H
  #include <stdlib.h>
#endif

#ifndef _TIME_H
  #include <time.h>
#endif

#ifndef _UNISTD_H
  #include <unistd.h>
#endif


/* --------- strftime_ms() ----------------------------- */
/*
  Funktion wie strftime(), bei Bedarf mit Millisekunden oder Mikrosekunden,
  wenn statt des Format-Zeichens %S  die Sequenz %mS (Millisekunden) oder
  %us (Mikrosekunden) geschrieben wird.
  Bei %mS werden die Sekunden statt 99 als 99.999 ausgegeben, bei
  %uS in 99.999999.

  strftime_ms() besitzt gegenüber der Funktion strftime()
  einen weiteren Eingangsparameter  long usec.
  Mit diesem Parameter werden die Microsekunden übergeben, die in der
  Struktur struct tm *time_stru nicht übergeben werden.

  Wenn vor Aufruf der Funktion strftime_ms() eine der Funktionen
  get_iec_dateandtime(),get_iec_time() oder get_s7_dateandtime()
  aufgerufen wurde, dann ist der Ausgabewert tv->tv_usec
  ( struct timeval tv; ) der geeignete Eingangsparameter.

-------
```

```
  strftime_ms() works like strftime, but has 2 additional formats to print
  seconds.
  "%mS" (instead of %S) prints also the broken down milliseconds 99.999
  "%uS" (instead of %S) prints also the broken down microseconds 99.999999

  In comparison to strftime() the additional input parameter,
  "long usec", is used. This parameter contains the micro seconds, which are
  not contained in the structure struct tm *time_stru.

  If one of the functions get_iec_dateandtime(),get_iec_time() or
  get_s7_dateandtime() are used prior to strftime_ms(), their output
  parameter tv->tv_usec ( struct timeval tv; )  is the best choice to
  supply usec.

*/

char* strftime_ms(char dstr[],size_t dstr_len, const char format[], \
      const struct tm *time_stru, long usec )
;
```