



**Emig** Software-Entwicklung und Vertrieb  
Engelstraße 17  
66564 Ottweiler-Fürth, Tel.+Fax 06858/6197

Programmpaket

**convert\_lib**

Version V2.1 (16.11.2002)

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

convert\_lib wurde entwickelt von



**HEISCH AUTOMATISIERUNGSTECHNIK**

Ingenieurbüro für Industrieautomatisierung

Ostring 15 Landau/Pfalz Tel. 6341 / 890-117 FAX -118

Postadresse: Im Vorderen Großthal 4 76857 Albersweiler / Pfalz

<http://www.emig-software.de>

<http://sites.inka.de/heisch>

# INHALT

## 1. ALLGEMEINES

- 1.1. Warenzeichen
- 1.2. Unterstützte Betriebssysteme und Systemanforderungen
- 1.3. Änderungen ( **WICHTIG für upgrades**)
- 1.4. Lieferumfang
- 1.5. Fehlende Formate und Upgrades
- 1.6. Haftung

## 2. Überblick über die vorhandenen Funktionen

- 2.1. Integer und BCD-Formate
- 2.2. Gleitpunkt-Formate
- 2.3. Datum-Uhrzeit-Formate
- 2.4. String-Formate
- 2.5. Hilfsfunktionen Strings

## **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.

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.

convert\_lib wurde unter Linux entwickelt, die meisten Teile davon sollen aber sogar unter MS-Windows laufen. ( So wurden z.B. die Gleitpunkt-Funktionen in MS-Windows basierten Programmen eingesetzt, es wurde mit dem Borland C-Builder(R) compiliert. )

### **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 unter S.u.S.E-Linux 6.3 entwickelt, sollte aber auch auf jeder anderen Linux-Distribution laufen, sofern sie das ELF-Format unterstützt und auf der glibc basiert.

Weil convert\_lib in Quellcode ausgeliefert wird, sollte es auf jedem 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)

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 aller Funktionen
s5types.c	Alle Funktionen zusammen.
s5types_i.c	die Integer und BCD-Funktionen
s5types_fp.c	die Gleitpunkt-Funktionen
s5types_t.c	die Datum-Uhrzeit-Funktionen
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

convert\_lib ist in Entwicklung.

Die Bibliothek ist nicht komplett, d.h. es sind noch nicht alle Formate realisiert, die es in Simatic S5 und S7 gibt.

Wir garantieren nicht, daß die noch fehlenden Formate in Zukunft unterstützt werden.

convert\_lib wird so ausgeliefert, wie es momentan ( siehe 2.) ist.

**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. Überblick über die vorhandenen Funktionen

### Inhalt von s5types.h

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

## 2.1 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)
;
```

## 2.2 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)
;

```

## 2.3 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 normally 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 ) Therefore 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 controlled 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 Sommerzeit
            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 since 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 Zeitzone-Korrektur nicht möglich.
Das Datum wird weitergegeben wie es ist.

----

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

```

because there is no time-of-day information it is impossible to execute a timezone correction. Therefore 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)



```

/* ----- 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)  
;
```

## 2.4. 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 gelesene 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)
;

```

## 2.5 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
***** */

#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
wenn statt des Format-Zeichens %S die Sequenz %mS geschrieben
wird, werden die Sekunden statt 99 als 99.999 ausgegeben

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 is the format token "%mS"
is used instead of "%S", the time is written with milliseconds
as a fraction of the seconds. (99 -> 99.999)

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 )
;

```