Referenz Handbuch

E-LAB AVRco

Pascal Multi-Tasking für Single Chips

Version für

AVR

Blaise Pascal Mathematiker 1623-1662

18.11.2012

Autor Rolf Hofmann
Editor Gunter Baab

E-LAB Computers

Mikroprozessor-Technik
Industrie-Elektronik
Hard + Software
8-Bit • 16-Bit • 32-Bit

E-LAB Computers
Grombacherstr. 27
D74906 Bad Rappenau
Tel 07268/9124-0
Fax 07268/9124-24
http://www.e-lab.de
info@e-lab.de

Wichtige Information

Weltweit wird versucht fehlerfreie Software herzustellen. Die Betonung liegt dabei auf versucht, denn es besteht eine einhellige Meinung, je komplexer eine Software ist, desto grösser die Wahrscheinlichkeit, dass Fehler eingebaut sind.

Wir sind aber nicht der Meinung, dass das ein Grundgesetz ist, und dass man deshalb mit Fehlern und Problemen einfach leben muss (obwohl das bei manchen Software Giganten offensichtlich so ist J).

Sollten Sie Fehler feststellen, so wären wir dankbar für jede Information darüber. Wir werden uns bemühen, dieses Problem möglichst kurzfristig zu lösen.

Es ist ebenfalls internationaler Konsens, dass für Folgekosten, die aus fehlerhafter Software entstehen, der Software Hersteller jedwede Haftung ausschliesst, es sei denn es wurde etwas anderes extra vereinbart.

Mit der Benutzung jeglicher Software Produkte von E-LAB Computers schliessen wir als Hersteller sämtliche Haftung aus daraus entstehenden Kosten bei Fehlern der Software aus.

Sie als Anwender bzw. Benutzer der Software erklären Sich damit einverstanden. Sollte das nicht der Fall sein, so dürfen Sie die Software auch nicht benutzen, bzw. einsetzen.

Wie gesagt, dieser Haftungsausschluss ist international Standard und üblich.

Dieses Handbuch und die zugehörige Software ist geistiges Eigentum von E-LAB Computers und damit urheberrechtlich geschützt. Diese Produkte werden dem Erwerber zur Nutzung überlassen. Der Erwerber darf diese Produkte nicht an dritte weitergeben noch weiterveräussern. Weitergabe von Kopien dieser Produkte an Dritte, ob gegen Endgeld oder nicht, ist ausdrücklich untersagt.

Wir meinen dass Sie, als Benutzer der Software, damit Geld verdienen können und damit auch eine Pflege der Produkte erwarten. Ein Produkt, das fast ausschliesslich aus Raubkopien besteht, bringt dem Hersteller/Autor kein Geld ein. Und damit kann ein Produkt auch nicht gepflegt und weiterentwickelt werden.

Es liegt also auch im Interesse des Anwenders, dass das Urheberrecht beachtet wird.

Das wars der Autor

1 AVRco Hilfe

1.1 Übersicht Compiler Schalter

Compiler Schalter

$ANALYSIS_ON
$BDATA
$BootApplication
$BOOTRST
$CodeStart
$D
$DATA
$DEBDELAY
$DEFINE
$DEPHASE
$DEVICE
$EEPROM
$EEPROM1
$ELSE
$ELSIF
$ELSIFDEF
$ENDIF
$ENUMTOASM
$HEXNAME
$HEXPATH
$I
$IDATA
$IDATA1
$IF
$IFDEF
$IFEND
$IFNDEF
$J
$LCDNOINIT
$LCDNOWAIT
$MODBUS
$NOADDRCHECK
$NOFRAME
$NOINIT
$NOOVRCHECK
$NORAMCHECK
$NOREGSAVE
$NORETURNCHECK
$NOSAVE
$NOSHADOW
$NOWATCHDOGAUTO
$OPTIMISE
$OPTI ALLOW_INLINE
$OPTI_BETA_OFF
$OPTI CHECK_RETURN_REGS
$OPTI NO_ALLOW_INLINE
$OPTI NO_CHECK_RETURN_REGS
$OPTI NO_CSE_OPT
$OPTI_QUICK
$OPTI SMARTLINK_ONLY
$OVERLAY
$PCU
$PDATA
$PHASE
$Q
$REUTILIZE
$SHOWERROR
$SHOWWARNING
$SL
$TYPEDCONST
$UDATA
$UNDEF
$VALIDATE
$VALIDATE_ALL
$VALIDATE_OFF
$VALIDATE_ON
$VectTab
$W
$WG
$X
$XDATA
$XIO
$ZEROLOCVARS

1.2 Übersicht Diverse Funktionen

Diverse Funktionen

AddAVFilter
BitCountOf
DeclAVfilter
FlashDownLoader
FlashLoaderExit
FlashLoaderInit
FlashLoaderRecv
FlashLoaderTransm
FloatAsLong
FreeMem
GetAdc
GetAVfilter
GetLargestBlock
GetMem
GetMemAvail
InterPolX
InterPolY
isSysTimerZero
I2CexpStat
LongAsFloat
PresetAVfilter
Read
ReadLn
ResetSysTimer
SetAdcFixed
SetAVfilter
SetDeviceLock
SetServoChan
SetServoOffs
SetSysTimer
SetSysTimerM
SizeOf
SwapIPaddr
SwapMACaddr
TestDeviceLock
Trap
WatchDogStart
WatchDogStop
WatchDogTrig
Write
WriteLn

1.3 Übersicht Fix64 Funktionen

der Fix64 Typ muss imortiert werden: **from** system **import** ..., Fix64;
Folgende komlexe Fix64 Funktionen müssen mit ***uses*** *uFix64* eingebunden werden.
Die grundlegende Funktionen sind Teil des Compilers und bei den Typen der entsprechenden Funktionen in Übersicht mathematische Funktionen 5 erwähnt.

Fix64ArcCos
Fix64ArcCosD
Fix64ArcCosh
Fix64ArcCot
Fix64ArcCotD
Fix64ArcCsc
Fix64ArcCscD
Fix64ArcSec
Fix64ArcSecD
Fix64ArcSin
Fix64ArcSinD
Fix64ArcSinh
Fix64ArcTan
Fix64ArcTanD
Fix64ArcTan2
Fix64ArcTan2D
Fix64ArcTanh
Fix64Cos
Fix64CosD
Fix64Cosh
Fix64Cot
Fix64CotD
Fix64Csc
Fix64CscD
Fix64DegToRad
Fix64DegToRadD
Fix64DivInt
Fix64DivLong
Fix64Even
Fix64Exp
Fix64Integrate
Fix64IsPowOfTwo
Fix64Ln
Fix64Log
Fix64Log10
Fix64LogN
Fix64Mod
Fix64ModInt
Fix64MulInt
Fix64MulLong
Fix64Odd
Fix64Power
Fix64PowerInt
Fix64Quadrant
Fix64RadToDeg
Fix64Sec
Fix64SecD
Fix64Sin
Fix64SinD
Fix64Sinh
Fix64Sqrt
Fix64Tan
Fix64TanD
Fix64Tanh
Fix64ToFloat
Fix64ToHex
Fix64ToInt
Fix64ToLongInt
Fix64ToStr
Fix64ValueInTolerance
Fix64ValueInToleranceP
FloatToFix64
IntToFix64
StrToFix64

1.4 Übersicht mathematische Funktionen

Mathematische Funktionen

Abs
ArcTan
BCDtoByte
ByteToBCD
CalcCheckSum
Cos
CosD
CosInt
CosInt16
Dec
DecToLim
DecToLimWrap
DegToRad
Even
Exp
Frac
Hi
Higher
HiNibble
HiWord
Inc
IncToLim
IncToLimWrap
Int
IntegrateB
IntegrateI
IntegrateI8
IntegrateW
IntToFix64
IsPowOfTwo
Lo
LogN
Log10
LoNibble
Lower
LoWord
LowPassFW
Max
Min
Mirror16
Mirror32
Mirror8
MulDivByte
MulDivInt
MulDivInt8
MulDivLong
Negate
Odd
Ord
Parity
Pow
Pow10
Pred
RadToDeg
RotatePntI
Round
Sgn
Sign
Sin
SinD
SinInt
SinInt16
SizeOf
SquareDivInt
SquareDivInt8
Sqr
Sqr
Sqrt
Succ
Swap
SwapLong
Tan
TanD
Trunc
ValueInRange
ValueInTolerance
ValueInToleranceP
ValueTrimLimit
Within
WordToBCD

1.5 Übersicht MultiTasking Funktionen

Multi Tasking Funktionen

ClearDeviceLock
DecSema
GetCurProcess
GetPriority
GetProcessState
GetProcessID
IncSema
IsCurProcess
Lock
Priority
ProcWaitFlag
ResetProcess
Resume
ResumeAll
Schedule
SchedulerOff
SchedulerOn
SemaStat
SetDeviceLock
SetSema
Start_Processes
Suspend
SuspendAll
TestDeviceLock
UnLock
WaitPipe
WaitSema
WaitDeviceFree

1.6 Übersicht Schlüsselwörter

reservierte Schlüsselwörter

absolute
and
array
asm
assign
at
begin
Bit
bitset
boolean
break
by
byte
case
char
chr
class
close
const
continue
define
definefrom
define_fuses
define_usr
device
devicelock
div
do
downto
else
elsif
els_if
end
endasm
endcase
endfor
endif
endloop
endtry
endwhile
endwith
end_asm
end_case
end_for
end_if
end_loop
end_try
end_while
end_with
eof
eoln
except
exec
exitloop
exit_loop
export
false
file
finalization
float
for
forward
from
function
goto
idle
if
implementation
import
in
inherit
inherited
initialization
integer
interface
interrupt
int64
int8
label
locked
longint
longword
loop
mod
nil
not
object
of
or
override
pidcontrol
pipe
pointer
procedure
process
program
record
repeat
return
rol
ror
semaphore
sendsema
set
shl
shla
shortint
shr
shra
str
string
structconst
systimer
systimer32
systimer8
table
task
then
to
true
try
type
unit
until
userdevice
uses
using
val
var
variant
while
with
word
word64
xor

1.7 Übersicht String Funktionen

String Funktionen

Append
ArrToStr
BoolToStr
ByteToBin
ByteToHex
ByteToStr
Copy
Delete
ExtractFileExt
ExtractFileName
ExtractFilePath
FloatToStr
Fix64ToHex
Fix64ToStr
HexToInt
Insert
IntToBin
IntToHex
IntToStr
IPtoStr
Length
LongToHex
LongToStr
Long64ToHex
Long64ToStr
LowCase
LowerCase
MACtoStr
PadLeft
PadRight
Pos
PosN
SetLength
StrClean
StrReplace
StrToArr
StrToFix64
StrToFloat
StrToInt
StrToIP
StrtoMAC
Trim
TrimLeft
TrimRight
UpCase
UpperCase

1.8 Übersicht System Funktionen

System Funktionen und Prozeduren

Addr	DecSema	mDelay	RunTimeErr
Bit	DisableInts	NoInts	sDelay
BootRestart	Disable_JTAGport	NOP	SemaStat
Boot_Init	EEpromPtr	OnADCread	SetBit
CalcFlashCheck	EnableInts	OnFAT16_SS	SetSema
CalcFlashCheck_A	EnableIntsX	OnIdleProcess	SetTable
CalcFlashCheck_B	Enable_JTAGport	OnSchedulerEntry	SetVectTabBoot
CalcFlashCheck_S	Excl	OnSchedulerExit	Sleep
CheckFrameValid	FillBlock	OnSysTick	Start_Processes
CheckStackValid	FillRandom	OnTickTimer	System_Init
ClearRunErr	FlashPtr	OnTINA_SS	System_MCUCR_Init
CompareBlock	GetExceptResult	PopAllRegs	System_Reset
CompareIP	GetFrameFree	PopRegs	System_ShutDown
CompareNet	GetStackFree	PushAllRegs	SysTickDisable
CompareMAC	GetSysTimer	PushRegs	SysTickEnable
CopyBlock	GetTable	PowerSave	SysTickRestart
CPUsleep	GetTaskFrameFree	RaiseException	SysTickStop
CRCcheck	GetTaskStackFree	Random	Toggle
CRCstreamAdd	GetWatchDogFlag	RandomRange	uDelay
CRCstreamAddP	Incl	RestoreInts	uDelay_1
CRCstreamInit	IncSema	RunErr	UsrDevPtr

1.9 8bit File System

Import *SysTick, FileSystem, ...;*

From *System* ***Import*** *longWord, ...;*

Define

```
FileBuffer   = iData;
FileHandles  = 2, iData;
Disk_A       = 1024, readOnly;   {kBytes}
SecTrk_A     = 2;                {2sect/track = 512bytes/track}
TRKOFFS_A    = 1;                {1 reserved system track}
Disk_B       = 2048;             {kBytes}
SecTrk_B     = 32;               {32sect/track = 4096bytes/track}
```

Funktionen und Prozeduren

ChangeDir	FileClose	FileOpen	FileSize
DiskFree	FileCreate	FilePos	FileSysReset
DiskFormat	FileDelete	FileRead	FileWrite
DiskReset	FileExists	FileRename	GetCurDir
DiskSelect	FileFirst	FileReset	GetCurDisk
EndOfFile	FileGetAttr	FileRewrite	
FileAppend	FileHandleCheck	FileSeek	
FileChangeDir	FileNext	FileSetAttr	

1.10 ADC Port

```
Import SysTick, ADCPort;

Define
  ProcClock     = 4000000;          {4Mhz clock }
  SysTick       = 10;               {10msec Tick}
  ADCchans      = 2, iData;         {Kanäle 1 und 2 benutzen}
  //ADCchans    = 2, iData, int2;   {2-fach integrieren}
  //ADCchans    = [4, 7], iData;    {Kanäle 4 und 7 benutzen}
  //ADCchans    = [2..5], iData;    {Kanäle 2, 3, 4, 5 benutzen}
  //ADCchans    = [5], iData;       {nur Kanal 5 benutzen}
  ADCpresc      = 16;               {Vorteiler 16}
```

Achtung: der AVRco zählt die Kanäle 1, 2, 3, ... (entsprechend 0, 1, 2, ... beim Controller)

Funktionen und Prozeduren

GetAdc OnADCread SetAdcFixed

1.11 AVR CAN

```
Import SysTick, CAN_AVR ;

From SysTick import SystemTime16;    // or SystemTime32 (optional)

Define
  ProcClock       = 16000000;                    {16Mhz clock}
  CAN_AVR         = 16, 16, iData;               {RxPipe, TxPipe}
//  CAN_AVR       = 16, 16, iData, CAN_SysTime;  // RxPipe, TxPipe, memory,
                                                 // Systemtime (optional)
  CAN_AVRbaud     = CAN_Baud1000;

type
  tCANMessage11 = record
                    MOBIdx   : byte;          // Message Object = Maibox number
                    MsgIdent : word;          // Identifier ~ Accept mask 11bits
                    DLC      : byte;          // data length 0..8
                    data : array [0..7] of byte;
                  end;

  tCAN_baud       = (CAN_Baud25, CAN_Baud50, CAN_Baud100, CAN_Baud125,
                     CAN_Baud200, CAN_Baud250, CAN_Baud500, CAN_Baud1000);
  tAVR_CAN_Stat   = (CAN_ACKerror, CAN_FrameError, CAN_CRCerror, CAN_StuffError,
                     CAN_BitError, CAN_RxOK, CAN_TxOK, CAN_DLCwarn);
  tAVR_CAN_States = BitSet of tAVR_CAN_Stat;

var
  CAN_RxPipe : Pipe [defined] of tCANMessage11;
  CAN_TxPipe : Pipe [defined] of tCANMessage11;
```

Funktionen und Prozeduren

AVR_CAN_BaudRate	AVR_CAN_GetStatus	AVR_CAN_StartMessage
AVR_CAN_Disable	AVR_CAN_Init	AVR_CAN_RxErrCount
AVR_CAN_Enable	AVR_CAN_SetRxEMask	AVR_CAN_TxErrCount
AVR_CAN_GetError	AVR_CAN_SetRxMask	

1.12 Banking Port

```
Import SysTick, BankPort;

Define
  ProcClock     = 8000000;      {Hertz}
  SysTick       = 10;           {msec}
  StackSize     = $0030, iData;
  FrameSize     = $0030, iData;
  BankPort      = 3;            {Bank Ports}

{$BDATA 0}
var
  bb0 : byte;
  ww0 : word;

...

UserDevice BankDevIni;          // Bank Device init
begin
  (* is called at System Init Time *)
  (* initialize Device Hardware *)
end

UserDevice BankDevInp (bank : byte; adr : word) : byte;
begin
  ...
  return(xxx);
end;

UserDevice BankDevOut (bank : byte; adr : word; arg : byte);
begin
  ...
end;
```

Funktionen und Prozeduren

BankDevPtr	GetBankNum

1.13 BeepPort

```
Import SysTick, BeepPort, ..;

Define
  ProcClock     = 8000000;      {Hertz}
  SysTick       = 10;           {msec}
  StackSize     = $0020, iData;
  FrameSize     = $0030, iData;
  BeepPort      = PortB, 0;
```

Funktionen und Prozeduren

BeepChirpH BeepClick BeepOutErr BeepOutLH BeepStepHL
BeepChirpL BeepOut BeepOutHL BeepSiren BeepStepLH

1.14 DA Converter (XMega)

```
Import SysTick, DAC_A, DAC_B;

Define
  // The XMegas don't provide any Oscillator fuses.
  // So the application must setup the desired values
  // possible OSC types: extXTAL, extClock, ext32kHz, int32Khz, int2MHz, int32MHz

  //>> CPU=32MHz, PeripherX4=32MHz, PeripherX2=32MHz
  OSCtype     = int32MHz, PLLmul=4, prescB=1, prescC=1;
  SysTick     = 10;                 {msec}
  StackSize   = $0032, iData;
  FrameSize   = $0064, iData;
  DAC_A       = chan01, REF100;     // DAC_A channel 0 + 1 used
  DAC_B       = chan0, REFextB;     // DAC_B channel 0 used
```

Funktionen und Prozeduren

SetDacA SetDacB

1.15 DCF-77 Encoder

```
Import SysTick, DCFclock, ..;

Define
  ProcClock      = 8000000;          {Hertz}
  SysTick        = 10 , Timer2;      {msec}
  DCFclock       = iData;
  DCFport        = PinD, 2, negative; {Port, bitnummer, Polarität}
  DCFfieldMode   = reset;            // decrement, optional "reset"
```

Funktionen und Prozeduren

DCFDayLightSave DCFfield DCFready DCFupdate

1.16 DDS10 Synthesizer

```
Import SysTick, DDS10;          // and SPIdriver if necessary

Define ProcClock      = 16000000;   {16Mhz clock }
  DDS10Timer          = Timer1;     {use a 16bit Timer}
  DDS10port           = PortA;
  // DDS10port        = SPI;
  // DDS10port        = UserPort ;  {use DDS10IOS}
  DDS10Tables         = 1;          {use 2 lookup tables}

XMega
  DDS10Timer = Timer_C1;            {Timer_C0,_C1,_D0,_D1,_E0,_E1,_F0,_F1}
  DDS10port = SPI_C, SPImode3, SPImsb, PortF, 4; {Mode 0..3, MSB/LSB, SS-Port, SS-Pin}

type
  tdsMode = (dsSine, dsTriaLeft, dsTriaSym, dsTriaRight, dsSquare);
```

Funktionen und Prozeduren

DDS10buildTab DDS10setFrequ DDS10setTab DDS10start DDS10stop

1.17 FAT16 File System

Import *SysTick, FAT16, ...;*

From *System* ***Import*** *longword, ...;*

Define

```
ProcClock       = 16000000;          {Hertz}
SysTick         = 10;                {msec}
StackSize       = $0040, iData;
FrameSize       = $0100, iData;
FAT16           = MMC_SPI, iData;
F16_MMCspeed    = standard;          // standard, slow, fast
F16_FileHandles = 4;
F16_DirLevels   = 2;
```

siehe auch ExtractFileExt, ExtractFileName, ExtractFilePath

Funktionen und Prozeduren

F16_BlockRandomWrite	F16_FileAppend	F16_FileReset	F16_GetUsedHandles
F16_BlockRead	F16_FileAssign	F16_FileSeek	F16_PathExist
F16_BlockWrite	F16_FileClose	F16_FileSetAttr	F16_PathExpand
F16_ChangeDir	F16_FileCreate	F16_FileSetDate	F16_RandomWrite
F16_CheckDisk	F16_FileCopy	F16_FileSize	F16_ReadSector
F16_CheckHandle	F16_FileDelete	F16_FileSizeH	F16_RemoveDir
F16_CreateDir	F16_FileExist	F16_FindFirst	F16_StrToDate
F16_DateToStr	F16_FileGetAttr	F16_FindNext	F16_StrToTime
F16_DirGetDate	F16_FileGetDate	F16_GetCurDir	F16_TimeToStr
F16_DiskInit	F16_FilePos	F16_GetDiskError	F16_WriteSector
F16_DiskReset	F16_FileRename	F16_GetDiskFree	
F16_EndOfFile	F16_FileRewrite	F16_GetDiskSize	

1.18 Flash Writer

Import *SysTick, FlashWrite, ..;*

Funktionen und Prozeduren

FlashClearPage	FlashErasePage	FlashReadFuses	FlashWritePage
FlashCopyF2R	FlashInitPage	FlashReadPage	
FlashCopyR2F	FlashProgPage	FlashWriteFuses	

1.19 Frequency Counter/Timer

Import *SysTick, FreqCount [, FreqCount2];*

Define

```
ProcClock   = 8000000;   {8Mhz clock }
SysTick     = 2.0;       {2msec Tick}
FreqTimer   = Timer1;    (used 16bit Timer}
FreqTimer2  = Timer3;    (restricted to Timer3}
```

Type

tFreqCountMode = (TFreqBaseNone, TFreqBase100Hz, TFreqBase1kHz, TFreqBase10kHz, TFreqBase100kHz, TFreqBase1MHz, TTimeBase100s, TTimeBase10s, TTimeBase1s, TTimeBase100ms, TPulseBase100s, TPulseBase10s, TPulseBase1s, TPulseBase100ms);

Funktionen und Prozeduren

FreqCountRestart	GetFreqCountMode	GetTimeCounterP
FreqCountRestart2	GetFreqCountMode2	GetTimeCounterP2
GetFreqCounter	GetFreqCountOvrFlow	SetFreqCountMode
GetFreqCounterL	GetFreqCountOvrFlow2	SetFreqCountMode2
GetFreqCounter2	GetTimeCounter	
GetFreqCounter2L	GetTimeCounter2	

1.20 I2C Disp7

```
Import I2Cport, I2C_Disp7;
or
Import TWImaster, I2C_Disp7;
or
Import TWInet, I2C_Disp7;              // use Master mode

I2Cport:
Define
  ProcClock     = 8000000;             {8Mhz clock }
  I2Cport       = PortC;               {port used}
  I2Cdat        = 7;                   {bit7-PortC}
  I2Cclk        = 6, 4;                {bit6-PortC, optional delay 4}
  I2C_Disp7     = I2C_Soft, iData;     {use Software I2Cport, buffer loc}
  // use 2 Displays
  I2C_7sDig1    = 8;                   {first display 8digits}
  I2C_7sDig2    = 6;                   {second display 8digits}
  I2C_7Mode     = wrap;

TWImaster:
Define
  ProcClock     = 8000000;             {8Mhz clock }
  TWIpresc      = TWI_BR100;           {100kBit/sec alt. TWI_BR400}
  I2C_Disp7     = I2C_TWI, xData;      {use TWIport, buffer location}
  // use 4 Displays
  I2C_7sDig1    = 4;                   {first display 4digits}
  I2C_7sDig2    = 4;                   {second display 4digits}
  I2C_7sDig3    = 4;                   {third display 4digits}
  I2C_7sDig4    = 4;                   {fourth display 4digits}
  I2C_7Mode     = shiftLeft;

TWInetMaster:
Define
  ProcClock     = 8000000;             {8Mhz clock }
  TWInode       = 05;                  {default address in slave mode}
  TWIpresc      = TWI_BR400;           {400kBit/sec alt. TWI_BR100}
  TWIframe      = 4, iData;            {buffer/packet size}
  TWIframeBC    = 6;                   {option broadcast buffer/packet size}
  TWInetMode    = Master;
  I2C_Disp7     = I2C_TWI, iData;      {use TWIport, buffer location}
  // use 1 Display
```

```
I2C_7sDig1   = 4;                {4digits}
I2C_7Mode    = wrap;
```

Type

```
TI2C_DISP7   = (Disp7_1, Disp7_2, Disp7_3, Disp7_4);
TI2C_Ctrl7   = (Disp7_On, Disp7_Off, Disp7_Test);
```

Var

I2C_7Buff1, I2C_7Buff2, I2C_7Buff3, I2C_7Buff4;

Funktionen und Prozeduren

I2C_Disp7Clear	I2C_Disp7DigitBlink	I2C_Disp7Init	I2C_Disp7Refresh
I2C_Disp7ClEOL	I2C_Disp7Dim	I2C_Disp7Out	I2C_Disp7Set
I2C_Disp7Ctrl	I2C_Disp7Get	I2C_Disp7Pos	

1.21 I2C Port

Import *SysTick, I2Cport;*

Define

```
ProcClock    = 4000000;     {4Mhz clock }
SysTick      = 10;          {10msec Tick}
I2Cport      = PortB;       {use port B}
I2Cclk       = 0;           {clock-pin = port B bit 0}
//I2Cclk     = 0,NOPs
I2Cdat       = 3;           {data-pin = port B bit 3}
```

Funktionen und Prozeduren

I2Cinp I2Cout I2Cstat

1.22 I2Cexpand

Import *I2Cport, I2Cexpand;*
or
Import *TWImaster, I2Cexpand;*
or
Import *TWInet, I2Cexpand;* *// use Master mode*
or ***XMega***
Import *TWI_C, I2Cexpand;* *// use TWI_C, TWI_D, TWI_E or TWI_F*

I2Cport:
Define

```
ProcClock    = 8000000;     {8Mhz clock }
I2Cport      = PortC;       {port used}
I2Cdat       = 7;           {bit7-PortC}
I2Cclk       = 6, 4;        {bit6-PortC, optional delay 4}
I2Cexpand    = I2C_Soft, $38;   {use Software I2Cport, 9554A}
I2CexpPorts  = Port0, Port4;    {use Port0 and Port4}
```

TWImaster:
Define

```
ProcClock    = 8000000;     {8Mhz clock }
TWIpresc     = TWI_BR100;   {100kBit/sec alt. TWI_BR400}
```

```
  I2Cexpand     = I2C_TWI, $38;        {use TWIport, 9554A}
  I2CexpPorts   = Port1, Port2;        {use Port1 and Port2}
```

TWInetMaster:

```
Define
  ProcClock     = 8000000;             {8Mhz clock }
  TWInode       = 05;                  {default address in slave mode}
  TWIpresc      = TWI_BR400;           {400kBit/sec alt. TWI_BR100}
  TWIframe      = 4, iData;            {buffer/packet size}
  TWIframeBC    = 6;                   {option broadcast buffer/packet size}
  TWInetMode    = Master;
  I2Cexpand     = I2C_TWI, $20;        {use TWIport, 9554}
  I2CexpPorts   = Port7;               {use Port7}
```

```
XMega
Define
  OSCtype       = int32MHz, PLLmul=4, prescB=1, prescC=1;
  TWIpresc      = TWI_BR100;           {100kBit/sec alt. TWI_BR400}
  I2Cexpand     = TWI_C, $38;          {use TWIportC, 9554A}
  I2CexpPorts   = Port1, Port2;        {use Port1 and Port2}

Var
  TWI_DevLock : DEVICELOCK;
  TWI_DevLockTN : DEVICELOCK;          // XMega TN = C, D, E or F
```

Funktionen und Prozeduren

I2CexpStat

1.23 I2Cexpand_5

```
Import I2Cport, I2Cexpand_5;
or
Import TWImaster, I2Cexpand_5;
or
Import TWInet, I2Cexpand_5;          // use Master mode
or XMega
Import TWI_C, I2Cexpand_5;           // use TWI_C, TWI_D, TWI_E or TWI_F
```

I2Cport:

```
Define
  ProcClock     = 8000000;             {8Mhz clock }
  I2Cport       = PortC;               {port used}
  I2Cdat        = 7;                   {bit7-PortC}
  I2Cclk        = 6, 4;                {bit6-PortC, optional delay 4}
  I2Cexpand_5   = I2C_Soft, $38;       {use Software I2Cport}
  I2CexpPorts_5= Port0, Port4;         {use Port00..04. and Port05..09 = 2x PCA9698}
```

TWImaster:

```
Define
  ProcClock     = 8000000;             {8Mhz clock }
  TWIpresc      = TWI_BR100;           {100kBit/sec alt. TWI_BR400}
  I2Cexpand_5 = I2C_TWI, $18;          {use TWIport}
  I2CexpPorts_5= Port00, Port05;       {use Port00..04. and Port05..09 = 2x PCA9698}
```

TWInetMaster:

```
Define
  ProcClock     = 8000000;          {8Mhz clock }
  TWInode       = 05;               {default address in slave mode}
  TWIpresc      = TWI_BR400;        {400kBit/sec alt. TWI_BR100}
  TWIframe      = 4, iData;         {buffer/packet size}
  TWIframeBC    = 6;                {option broadcast buffer/packet size}
  TWInetMode    = Master;
  I2Cexpand_5   = I2C_TWI, $20;     {use TWIport}
  I2CexpPorts_5= Port00, Port05;    {use Port00..04. and Port05..09 = 2x PCA9698}

XMega
Define
  OSCtype       = int32MHz, PLLmul=4, prescB=1, prescC=1;
  TWIpresc      = TWI_BR100;        {100kBit/sec alt. TWI_BR400}
  I2Cexpand_5   = TWI_C, $18;       {use TWIport C}
  I2CexpPorts_5= Port00, Port05;    {use Port00..04. and Port05..09 = 2x PCA9698}

Var
  TWI_DevLock : DEVICELOCK;
  TWI_DevLockTN : DEVICELOCK;       // XMega TN = C, D, E or F for TWI_C…TWI_F
```

Funktionen und Prozeduren

I2CexpStat_5

1.24 Increment Counter

```
Import SysTick, IncrPort, ..;

Define
  ProcClock     = 8000000;      {Hertz}
  SysTick       = 10;           {msec}
  StackSize     = $0010, iData;
  FrameSize     = $0010, iData;
  IncrCounter   = 16, 2;        {16 bit integer, 2 Phasen}
  IncrPort      = PinD, $C0;    {PinD, Portpin 6 + 7}
```

Funktionen und Prozeduren

ClearIncrementVal GetIncrementRel GetIncrementVal SetIncrementVal

1.25 Increment Counter 4chan

```
Import SysTick, IncrPort4, ..;

Define
  ProcClock     = 16000000;     {Hertz}
  StackSize     = $0020, iData;
  FrameSize     = $0040, iData;
  IncrPort4     = PinA, 2, 32;  // pin-reg used, channels, 16 or 32bit integer
  IncrScan4     = Timer3, 10;   // timer used, scan rate 10kHz (1..100)
```

Funktionen und Prozeduren

ClearIncrAll4 GetIncrRel4 IncrCount4start SetIncrVal4
ClearIncrVal4 GetIncrVal4 IncrCount4stop

1.26 IOexpand

Import *IOexpand;*

Define

ProcClock	*= 8000000;*	*{8Mhz clock }*
IOexpand	*= PortD, 2, iData;*	*{Port, Port-startbit, memory loc}*
IOexpInp	*= 4;*	*{4x8 = 32 bit input}*
IOexpOutp	*= 4;*	*{4x8 = 32 bit output}*

Var

```
IOexpInpArr : array [0.. IOexpInp-1] of byte;
IOexpInp0 : byte;
IOexpInp1 : byte;
IOexpInp2 : byte;
...
IOexpOutpArr : array[0.. IOexpOutp-1] of byte;
IOexpOutp0 : byte;
IOexpOutp1 : byte;
IOexpOutp2 : byte;
...
```

Funktionen und Prozeduren

IOexpUpdate

1.27 KeyBoard

Import *SysTick, MatrixPort, ... ;*

Option
From *MatrixPort* ***import*** *MatrixTimer;*

Define

MatrixRow	*= PortA, 4;*	*{use PortA, start with bit4}*
MatrixCol	*= PinA, 0;*	*{use PinA, start with bit0}*
MatrixType	*= 3, 4*	*{3 Rows at PortA, 4 Columns at PinA}*
Debounce	*= 4;*	*{optional debounce in systick counts)*

Option

MatrixPipe	*= PipeLen, FirstRepeat, RepeatRate;*

Type

Keys	*= (Key1, Key2, ..., KeyN);*
KeySet	*= BitSet* ***of*** *Keys*

Const

```
lastMatrixKey : Keys = KeyN;
```

Option
Var

```
MatrixKeyPipe: Pipe[PipeLen] of Keys;
```

Funktionen und Prozeduren

ClearKeyBoard	KeyBoardEnable	KeyStat	ReadKeyBoard
GetKey	KeyBoardRepeat	KeyStatRaised	
GetKeyRaised	KeyRaised	ReadKey	

1.28 KeyBoard8

```
Import SysTick, KeyPort8, ... ;

Option
From KeyPort8 import KeyBoardTimer;

Define
   KeyB8Row    = PortA, 2;      {use PortA, start with bit2}
   KeyB8Col    = PinD;          {use PinD complete input port}
   KeyB8Type   = 4;             {4 Rows at PortA, 8 Columns at PinD}
   Debounce    = 5;             {nn = SysTicks, optional}
Option
   KeyB8Pipe   = PipeLen, FirstRepeat, RepeatRate;

Type
   Keys        = (Key1, Key2, ..., KeyN);
   KeySet      = BitSet of Keys;

Const
   lastKeyBoardKey : Keys = KeyN;

Option
Var
   KeyboardPipe : Pipe [PipeLen] of Keys;
```

Funktionen und Prozeduren

ClearKeyBoard8	KeyBoardEnable8	KeyStatRaised8	ReadKey8
GetKey8	KeyBoardRepeat8	KeyStat8	
GetKeyRaised8	KeyRaised8	ReadKeyBoard8	

1.29 LCD Bargraph

```
Define
   ProcClock        = 8000000;          {8Mhz clock }
   ...
   ...
   LCDBargraph1    = LCDmultiPort;
   LCDBargraph2    = LCDmultiPort;
   LCDBargraph3    = LCDmultiPort;
   LCDBargraph4    = LCDmultiPort;
   // LCDBargraph4 = LCDPort;
```

Funktionen und Prozeduren

LCDbarInit_M	LCDbarOut1	LCDbarOut3	LCDbarSet1	LCDbarSet3
LCDbarInit_P	LCDbarOut2	LCDbarOut4	LCDbarSet2	LCDbarSet4

1.30 LCD Display

Import *SysTick, LCDport;*

Define

```
ProcClock     = 4000000;     {4Mhz clock }
SysTick       = 10;          {10msec Tick}
LCDtype       = 44780;       {66712,0070, 0073}
LCDport       = PortA;       {Port Adresse}
LCDRows       = 2;           {2-Zeiliges Display}
LCDColumns    = 16;          {16-stelliges Display}
```

Option 1

```
LCDport = PortC, 2, PortA, 3;          // controlport, bit, Dataport, bit. Splitted ports
```

Option 2

```
LCDtype = 0073; {KS0073}
LCDport = SPI_Soft, PortB, 1, 2, 3, 0;    // PortX, SCK, MOSI, MISO, SS
```

Option 3

```
LCDtype = 0073;                        {KS0073}
LCDport = SPI;                         // Hardware SPI
LCDport = SPI_C, PortB, 1;             // Hardware SPI, SS_port, SS_pin XMega
```

Funktionen und Prozeduren

LCDbarInit_P	LCDbarSet1	LCDcharsetP	LCDcursor	LCDoff	LCDupper
LCDbarOut1	LCDbarSet2	LCDclr	LCDgetXY	LCDon	LCDxy
LCDbarOut2	LCDbarSet3	LCDclrEol	LCDhome	LCDout	
LCDbarOut3	LCDbarSet4	LCDclrLine	LCDinp	LCDsetup	
LCDbarOut4	LCDcharset	LCDctrl	LCDlower	LCDstat	

1.31 LCD Display Multi

Import *I2Cport, LCDmultiPort;*
or
Import *TWImaster, LCDmultiPort;*
or
Import *TWInet, LCDmultiPort;* *// use Master mode*
or ***XMega***
Import *TWI_C, TWI_E, LCDmultiPort;* *// TWI_C, TWI_D, TWI_E, TWI_F*

I2Cport:

Define

```
  ProcClock       = 8000000;     {8Mhz clock }
  I2Cport         = PortC;       {port used}
  I2Cdat          = 7;           {bit7-PortC}
  I2Cclk          = 6, 4;        {bit6-PortC, optional delay 4}
  LCDmultiPort    = I2C_Soft;    {use Software I2Cport}
  LCDTYPE_M       = 66712;       {LCD controller type}
  LCDrows_M       = 2;           {2 rows}
  LCDcolumns_M    = 20;          {20 chars per line}
```

TWImaster:

Define

```
  ProcClock       = 8000000;     {8Mhz clock }
```

```
  TWIpresc          = TWI_BR100;  {100kBit/sec alt. TWI_BR400}
  LCDmultiPort      = I2C_TWI;    {use TWIport}
  LCDTYPE_M         = 44780;      {LCD controller type}
  LCDrows_M         = 4;          {4 rows}
  LCDcolumns_M      = 16;         {16 chars per line}
```

TWInetMaster:

Define

```
  ProcClock         = 8000000;    {8Mhz clock }
  TWInode           = 05;         {default address in slave mode}
  TWIpresc          = TWI_BR400;  {400kBit/sec alt. TWI_BR100}
  TWIframe          = 4, iData;   {buffer/packet size}
  TWIframeBC        = 6;          {option broadcast buffer/packet size}
  TWInetMode        = Master;
  LCDmultiPort      = I2C_TWI;    {use TWIport}
  LCDTYPE_M         = 0070;       {LCD controller type}
  LCDrows_M         = 1;          {1 row}
  LCDcolumns_M      = 12;         {12 chars per line}
```

XMega

Define

```
  OSCtype           = int32MHz, PLLmul=4, prescB=1, prescC=1;
  TWIprescC         = TWI_BR100;  {100kBit/sec alt. TWI_BR400}
  LCDmultiPort      = I2C_TWI;    {use TWIport}
  LCDTYPE_M         = 44780;      {LCD controller type}
  LCDrows_M         = 4;          {4 rows}
  LCDcolumns_M      = 16;         {16 chars per line}
```

Type

```
  TLCD_num = (LCD_m1,LCD_m2,LCD_m3,LCD_m4,LCD_m5,LCD_m6,LCD_m7,LCD_m8);
```

Var

```
  TWI_DevLock : DEVICELOCK;
  TWI_DevLockTN : DEVICELOCK; // XMega, TN = C, D, E, F
```

Funktionen und Prozeduren

LCDbarInit_M	LCDbarSet2	LCDclrLine_M	LCDhome_M	LCDsetPort_M
LCDbarOut1	LCDbarSet3	LCDclr_M	LCDinp_M	LCDsetup_M
LCDbarOut2	LCDbarSet4	LCDctrl_M	LCDoff_M	LCDstat_M
LCDbarOut3	LCDcharSet_M	LCDcursor_M	LCDon_M	LCDxy_M
LCDbarOut4	LCDcharSet_MP	LCDgetPort_M	LCDout_M	
LCDbarSet1	LCDclrEOL_M	LCDgetXY_M	LCDportInp_M	

1.32 LCD Edit

Uses *FEdit;*

feste Konstanten:

```
EdEditLength    : Byte = 40
EdLabelLength   : Byte = 20
EdTimeDelim     : Char = ':'
EdDateDelim     : Char = '.'
EdIPDelim       : Char = '.'
```

änderbare Konstanten:

```
EdPreClearLine          : Boolean = True
EdBooleanTrue    : String= ' AN'
EdBooleanFalse          : String= 'AUS'
KBRepeatTrigger         : Byte = 100;          // in SysTicks
KBRepeatDelay           : Word = 100;          // in SysTicks
```

Typen:

```
Type   tEdArrayLocation = (edRam, edEEProm, edFlash);
Type   tEdLCDType = (edLCDnone, EdLCDStandard, EdLCDMulti);
Type   tEdKeys = (EdKeyNone, EdKeyUP, EdKeyDown, EdKeyLeft, EdKeyRight, EdKeyExit);
Type   TEdActEditor = (edNoneEd, edTimeEd, edDateEd, edByteEd, edBooleanEd, edStringEd,
         edIntegerEd, edIPAddressEd, edLongIntEd, edWordEd, edLongWordEd, edListEd);
Type   tEdErrorEvent = (edLeftLim, edRightLim, edUPLim, edDownLim, edNoKeyHandler,
         edNoLCDDefined);
Type   tEdKeyBoardHandler =
         Function (ActiveEditor : tEdActEditor; LookKey : tEdKeys) : tEdKeys;
Type   tEdErrorEventHandler =
         Procedure (ActiveEditor : tEdActEditor; ErrorEvent : tEdErrorEvent);
Type   tEdIPAddress =
       Record
         IPOct1,
         IPOct2,
         IPOct3,
         IPOct4 : Byte;
       end;
```

Standard Parameter der Editoren:

EdValue	Übergabe-Wert an die Funktion
LeadLabel	Optionales führendes Label des Edit-Feldes
Postlabel	Optionales Label nach dem Edit Feld z.B. *LeadLabel*-> Masse: 12.54 kg <- *PostLabel* ^- *EdValue*
X, Y	Koordinaten auf dem LCD-Display
BlinkCursor	Blinkender Block-Cursor auf dem LCD-Display an der Stelle Edit
VMin,VMax	Minimaler-Maximaler Edit-Wert
Repeater	AutoRepeater ein/aus. Für String und List editor
Decimal	Dezimalstellen des Wertes

Keyboard Handler:

```
Function MyKeyHandler (ActiveEditor : tEdActEditor; ReturnKey : tEDKeys) : tEDKeys;
begin
  case ActiveEditor of
    ...
```

Error Handler:

```
Procedure MyErrorEventHandler (ActiveEditor : tEdActEditor; ErrorCode: tEdErrorEvent);
begin
  case ActiveEditor of
    ...
```

Funktionen und Prozeduren

EdBoolean	EdInteger	EdLongInt	EdTime
EdByte	EdIPaddress	EdLongWord	EdWord
EdDate	EdList	EdString	

1.33 LCD Graphic

```
Import SysTick, LCDGraphic, ...;

// only if strings are used
From LCDGraphic Import CharSet;      {block CharSet, pixels}

Define
   LCDGraphic      = 240, 128, 8;    {x-pix, y-pix, accesswidth}
   GViewports      = 4, iData;       {logical ViewPorts, scalings}
   LCDgraphMode    = linear, iData;  {optional, linear is default}
or
   LCDgraphMode    = column, iData;  {column oriented controller}
or
   LCDgraphMode    = readonly, iData; {linear, readonly controller}

   DefCharSet      = 'Graphchars.pchr'; {FileName, stored into Flash}
or
   DefCharSet      = RAM;            {charset is stored into RAM}
   TGraphStr       = 20;             {Graphic Text String Length, max 24}

Type
   TGraphString    = String[n];      {max. 24 Zeichen}
   TWriteMode      = (wmClrPix, wmSetPix, wmXorPix);
   TtxtAlHor       = (alHorLeft, alHorCenter, alHorRight);
   TTxtAlVert      = (alVertBottom, alVertCenter, alVertTop);
   TTxtRotate      = (TxtRot0, TxtRot90, TxtRot180, TxtRot270);
   TTextBkGnd      = (bkNormal, bkTransp, bkInvers);
```

Funktionen und Prozeduren

gClearPixel	gDrawString	gSetTextMode	gSetCharSetRAM
gClearView	gDrawStringRel	gMoveTo	gSetLineColor
gClrScr	gFillCircle	gMoveToRel	gSetLineMode
gDispRefresh	gFillRect	gOpenView	gSetPixel
gDrawBitMap	gFrameView	gPntToScale	gSetTextBkGnd
gDrawBitMapN	gGetCurView	gRestoreView	gSetTextColor
gDrawCircle	gGetLineColor	gSaveViewv	gSetTextJustify
gDrawLine	gGetLineMode	gScaleToPnt	gGetTextMode
gDrawLineTo	gGetTextBkGnd	gScaleView	gSwitchView
gDrawLineToRel	gGetTextColor	gSetBitMapRAM	gXorPixel
gDrawRect	gGetTextJustify	gSetCharSet	

1.34 LED 14seg Display

```
Import SysTick, Disp14sPort;

Define
   ProcClock       = 8000000;        {8Mhz clock }
   SysTick         = 5;              {5msec}
   Disp14sPort     = PortB, 0;       {Port, start Portbit}
   Disp14Mode      = ShiftRight, Blank;
   Disp14Digits    = 6, iData;

Var Disp14Buff : array [0.. Disp14Digits-1] of word;
```

Funktionen und Prozeduren

Disp14Blink Disp14ClrEOL Disp14Out Disp14Test
Disp14Clear Disp14DigBlink Disp14Pos

1.35 LED 7seg Display

Import *SysTick, Disp7sPort;*

Define

```
ProcClock     = 4000000;              {4Mhz clock }
SysTick       = 10;                   {10msec Tick}
Disp7sPort    = PortA, Mux;           {Port Adresse, multiplexed}
//Disp7sPort  = PortA, NonMux;        {Port Adresse, nicht multiplexed}
//Disp7sPort  = PortA, Mux, startpin; {Port Adresse, multiplexed mit Startpin}
DispMode      = ShiftLeft;            {links durchschieben}
DispDigits    = 4, iData;             {4-stelliges Display}
DispMode      = ShiftLeft, noBlank;   {loptional}
```

Funktionen und Prozeduren

DispBlink DispClEOL DispOut Disp7Test
DispClear DispDigBlink DispPos

1.36 LED DOT Display

T.B.D

Funktionen und Prozeduren

LEDdotBlink LEDdotClr LEDdotDim LEDdotOn
LEDdotBlinkDigit LEDdotClrEOL LEDdotGetXY LEDdotOff
LEDdotCharSet LEDdotClrLine LEDdotHome LEDdotOut
LEDdotCharsetP LEDdotCursor LEDdotInit LEDdotXY

1.37 LPT Port

Import *SysTick, LPTport, ..;*

Define

```
ProcClock     = 8000000;      {Hertz}
SysTick       = 10;           {msec}
StackSize     = $0010, iData;
FrameSize     = $0010, iData;
LPTport       = PortA, PortB; // DataPort, ControlPort
```

or
Import *SysTick, TWImaster, LPTport, ..; // TWInetMaster is also possible*

Define

```
ProcClock     = 8000000;      {Hertz}
SysTick       = 10;           {msec}
StackSize     = $0010, iData;
```

```
  FrameSize     = $0010, iData;
  TWIpresc      = TWI_BR400;
  LPTport       = TWI_I2C, $24;  // $24 = TWIaddr
```

or

```
Import SysTick, I2Cport, LPTport, ..;
```

```
Define
  ProcClock     = 8000000;       {Hertz}
  SysTick       = 10;            {msec}
  StackSize     = $0010, iData;
  FrameSize     = $0010, iData;
  I2Cport       = PortA;
  I2Cclk        = 0;             // bit0, porta
  I2Cdat        = 1;             // bit1, porta
  LPTport       = Soft_I2C, $24; // $24 = I2Caddr
```

or ***XMega***

```
Import SysTick, TWI_C, LPTport, ..; // TWI_D, TWI_E, TWI_F are also possible
```

```
Define
  OSCtype       = int32MHz, PLLmul=4, prescB=1, prescC=1;
  SysTick       = 10;            {msec}
  StackSize     = $0020, iData;
  FrameSize     = $0040, iData;
  TWIprescC     = TWI_BR400;
  LPTport       = TWI_C, $24;    // $24 = TWIaddr
```

```
Type
  tLPTlines     = (lpStrobe, lpError, lpInit, lpSelect, lpACK, lpBusy, lpSelected, lpPaper);
  tLPTlineSet   = BitSet of tLPTlines;
```

Funktionen und Prozeduren

LPTctrl LPTdir LPTinit LPTinp LPTout LPTreset LPTstat

1.38 MIRF24 Port

```
Import SysTick…, MIRF24port, ..;
```

```
Define
  ProcClock     = 16000000; {Hertz}
  SysTick       = 10; {msec}
  StackSize     = $0040, iData;
  FrameSize     = $0040, iData;

  MIRF24port    = SPI_Soft, PortA, 2, 3, 4, 1, 0, 5;
                                  // SCK, MOSI, MISO, SS, CE, IRQ
// MIRF24port   = SPI, PortA, 0, 1, 2;          // standard SPI port
                                  // SS, CE, IRQ
//XMega: SPI_C, SPI_D, SPI_E, SPI_F

// MIRF24port   = MSPI_2, PortA, 0, 1, 2;       // MSPI_0..MSPI_3
                                  // SS, CE, IRQ …
//XMega: MSPI_C0, MSPI_C1, MSPI_D0, ...
```

```
uses uMIRF24;

Type
  tMRFchan    =  (mrfChan1, mrfChan2, mrfChan3, mrfChan4, mrfChan5,
                  mrfChan6, mrfChan7, mrfChan8, mrfChan9, mrfChan10,
                  mrfChan11, mrfChan12, mrfChan13, mrfChan14);
  tMRFpwr     =  (mrfdBm0, mrfdBm6, mrfdBm12, mrfdBm18);
  enMRFstat   =  (mrfTX_full, mrfRX_pn0, mrfRX_pn1, mrfRX_pn2,
                  mrfMAX_RT, mrfTX_DS, mrfRX_DR);
  tMRFstat    =  Bitset of enMRFstat;
  tMRFpkt     =  (mrfPKTnone, mrfPKTdata, mrfPKTbcast);
  tMRFrfSpeed =  (mrfRF250, mrfRF1000, mrfRF2000);
```

Funktionen und Prozeduren

MRFgetLostPkts	MRFgetState	MRFsetFreq	MRFsetRetryMax
MRFgetRetryCnt	MRFinit	MRFsetLocalAdr	MRFsetRetryTimeOut
MRFgetRxPower	MRFrxPacket	MRFsetPower	MRFsetRFspeed
MRFgetRxType	MRFsetChan	MRFsetPWRdown	MRFtxPacket

1.39 ModBus ASCII

```
Import SysTick, SerPort, ModBus;      //SerPort, SerPort2, SerPort3, SerPort4 are supported

From System Import Processes;         //driver is implemented in a process

Define                                //example for MEGA128
  ProcClock     = 16000000;                 //Hertz
  SysTick       = 10;                       //msec
  StackSize     = $0020, iData;
  FrameSize     = $0050, iData;
  Scheduler     = iData;
  SerPort       = 19200, Databit7, parEven, Stop1;   //ASCII default
  SerPortDTR    = PinB, 7, Positive;        //RF usually has busy signal
  SerCtrl       = PortD, 2, Positive;       //control line if RS485 used
  RxBuffer      = 255, iData;               //recommended, but may be lower
  TxBuffer      = 100, iData;               //recommended, but may be lower
  ModBus        = SerPort, 40;              //use port 1/2, capacity in words
  ModBusMode    = ASCII;                    //what modbus mode to use

Uses ModBusServASCII;         //modbus logic is in this unit

Type
  mb_InpB       = byte;       { !! always as couples }
  mb_RdWrB      = byte;       { !! always as couples }
  mb_InpW       = word;
  mb_RdWrW      = word;       {prefix mb_Inp is used to identify}
  mb_InpI       = Integer;    {that tag is read only, usually used}
  mb_RdWrI      = Integer;    {for mapping input from some sensor,}
  mb_InpW32     = longword;   {and prefix mb_RdWr is for read/write,}
  mb_RdWrW32    = longword;   {usually used for physical outputs.}
  mb_InpI32     = longInt;
  mb_RdWrI32    = longInt;    {sufixes B, W, I, W32, I32 and F are}
  mb_InpF       = Float;      {used to identify byte, word, int,}
  mb_RdWrF      = Float;      {longword, longint, and float types}
```

Funktionen und Prozeduren

mb_GetModBusDevID	mb_SetAfterRegisterRead	mb_SetBeforeRegisterRead
mb_GetModBusTimeout	mb_SetAfterRegisterWrite	mb_SetBeforeRegisterWrite
mb_SetAfterCoilRead	mb_SetBeforeCoilRead	mb_SetModBusDevID
mb_SetAfterCoilWrite	mb_SetBeforeCoilWrite	mb_SetModBusTimeout

1.40 ModBus RTU

```
Import SysTick, SerPort, ModBus;        //SerPort, SerPort2, SerPort3, SerPort4 are supported

From System Import Processes;           //driver is implemented in a process

Define                                  //example for MEGA128
  ProcClock     = 16000000;             //Hertz
  SysTick       = 10;                   //msec
  StackSize     = $0020, iData;
  FrameSize     = $0050, iData;
  Scheduler     = iData;
  SerPort       = 19200, Databit7, parEven, Stop1;   //ASCII default
  SerPortDTR    = PinB, 7, Positive;    //RF usually has busy signal
  SerCtrl       = PortD, 2, Positive;   //control line if RS485 used
  RxBuffer      = 255, iData;           //recommended, but may be lower
  TxBuffer      = 100, iData;           //recommended, but may be lower
  ModBus        = SerPort, 40;          //use port 1/2, capacity in words
  ModBusMode = RTU, Timer3;             //MODBUS mode, timer 1..3

Uses ModBusServRTU                      //MODBUS logic is in this unit

Type
  mb_InpB       = byte;         { !! always as couples }
  mb_RdWrB      = byte;         { !! always as couples }
  mb_InpW       = word;
  mb_RdWrW      = word;         {prefix mb_Inp is used to identify}
  mb_InpI       = Integer;      {that tag is read only, usually used}
  mb_RdWrI      = Integer;      {for mapping input from some sensor,}
  mb_InpW32     = longword;     {and prefix mb_RdWr is for read/write,}
  mb_RdWrW32 = longword;        {usually used for physical outputs.}
  mb_InpI32     = longInt;
  mb_RdWrI32 = longInt;         {sufixes B, W, I, W32, I32 and F are}
  mb_InpF       = Float;        {used to identify byte, word, int,}
  mb_RdWrF      = Float;        {longword, longint, and float types}
```

Funktionen und Prozeduren

mb_SetModBusDevID	mb_SetAfterRegisterRead	mb_SetBeforeRegisterWrite
mb_GetModBusExceptionStatus	mb_SetAfterRegisterWrite	mb_GetModBusDevID
mb_GetModBusTimeout	mb_SetBeforeCoilRead	mb_SetModBusExceptionStatus
mb_SetAfterCoilRead	mb_SetBeforeCoilWrite	mb_SetModBusTimeout
mb_SetAfterCoilWrite	mb_SetBeforeRegisterRead	

1.41 Pipes

Import *SysTick, ...*

From *System* ***Import*** *pipes, ...*

Funktionen und Prozeduren

PipeFlush	PipeRecv	PipeStat	WaitPipe
PipeFull	PipeRecv_ND	PipeSend	

1.42 Pulse Counter

Import *SysTick, PulseCount, {PulseCount2},..;*

Define

```
ProcClock      = 8000000;          {Hertz}
SysTick        = 10;               {msec}
StackSize      = $0010, iData;
FrameSize      = $0010, iData;
PulseCount     = Timer1;           {or Timer3..5 if present}
//PulseCount2  = Timer1;           {or Timer3..5 if present}
```

Funktionen und Prozeduren

GetPulseCount	PulseCountClear	PulseCountStart	PulseCountStop
GetPulseCount2	PulseCountClear2	PulseCountStart2	PulseCountStop2

1.43 PWM Port

PWMport1A, PWMport1B, PWMport1C, PWMport2A, -2B,
PWMport3A, -3B, -3C, PWMport4A, -4B, -4C, PWMport5A, -5B, -5C

Import *SysTick, PWMport1A, ..., PWMport3C, ...;*

Define

```
ProcClock      = 8000000;          {Hertz}
SysTick        = 10;               {msec}
StackSize      = $0030, iData;
FrameSize      = $0030, iData;
PWMpresc1      = $2;               {prescaler  timer1}
PWMres1        = $8;               {resolution timer1}
PWMmode1       = fast, negative    {optional define}
PWMpresc2      = $3;               {prescaler  timer1}
PWMres2        = $9;               {resolution timer1}
PWMmode2       = slow, negative    {optional define}
PWMpresc3      = $4;               {prescaler  timer3}
PWMres3        = $10;              {resolution timer3}
PWMmode3       = slow, positive    {optional define }
```

...

```
PWMport1A := ...
PWMport3B := ...
```

XMega:
PWMport_C0A, _C0B_, C0C, _C0D, _D0A, _D0B, _D0C, _D0D,
PWMport_E0A, _E0B_, E0C, _E0D, _F0A, _F0B, _F0C, _F0D,

PWMport_C1A, _C1B, _D1A, _D1B, _E1A, _E1B, _F1A, _F1B

Import *SysTick, PWM_C0A, PWM_C0B, PWM_C0C, PWM_C0D, PWM_C1A, PWM_C1B;*

Define

```
// The XMegas don't provide any Oscillator fuses.
// So the application must setup the desired values
// possible OSC types: extXTAL, extClock, ext32kHz, int32Khz, int2MHz, int32MHz

//>> CPU=32MHz, PeripherX4=32MHz, PeripherX2=32MHz
  OSCtype      = int32MHz,
  PLLmul=4,
  prescB=1,
  prescC=1;
  SysTick      = 10; {msec}
  StackSize            = $0064, iData;
  FrameSize            = $0064, iData;

  PWMpresc_C0      = $2;          // prescaler timerC0
  PWMres_C0        = 8;           // pwm resolution timerC0
  PWMpol_C0A       = negative;    // output polarity PWM_C0A
  PWMpol_C0B       = negative;    // output polarity PWM_C0B
  PWMpol_C0C       = positive;    // output polarity PWM_C0C
  PWMpol_C0D       = positive;    // output polarity PWM_C0D

  PWMpresc_C1      = $4;          // prescaler timerC1
  PWMres_C1        = 16;          // pwm resolution timerC1
  PWMpol_C1A       = positive;    // output polarity PWM_C1A
  PWMpol_C1B       = negative;    // output polarity PWM_C1A
```

Funktionen und Prozeduren XMega PWMPort_C0

EnablePWM_C0A SetPWM_C0A EnablePWM_C0B SetPWM_C0B
EnablePWM_C0C SetPWM_C0C EnablePWM_C0D SetPWM_C0D

Funktionen und Prozeduren
XMega PWMPort_D0A, PWMPort_D0B, PWMPort_D0C, PWMPort_D0D
XMega PWMPort_E0A, PWMPort_E0B, PWMPort_E0C, PWMPort_E0D
XMega PWMPort_F0A, PWMPort_F0B, PWMPort_F0C, PWMPort_F0D
analog zu PWMPort_C0A, PWMPort_C0B, PWMPort_C0C, PWMPort_C0D

Funktionen und Prozeduren XMega PWMPort_C1A

EnablePWM_C1A SetPWM_C1A EnablePWM_C1B SetPWM_C1BA

Funktionen und Prozeduren
XMega PWMPort_D1A, _D1B, PWMPort_E1A, _E1B, PWMPort_F1A, _F1B
analog zu PWMPort_C1A, PWMPort_C1B

1.44 RC5 Driver

Import *SysTick, RC5Rxport, ..;*
oder
Import *SysTick, RC5Txport, ..;*
oder

Import *SysTick, RC5Txport, RC5Rxport, ..;*

Defines für den Receiver

```
//  RC5RXPORT       = PinReg, PinNum, polarity     //polarity ist optional
    RC5mode         = rc_7bit;                     //rc_6bit = default
```

Defines für den Transmitter

```
//  RC5TXPORT       = Timer1, polarity, carrier;   //Timer und carrier Frequenz ist optional
    RC5TXPORT       = positive;
    RC5TXPORT       = negative;
    RC5TXPORT       = positive, 38;
    RC5TXPORT       = Timer1, positive;
    RC5TXPORT       = Timer3, negative, 36;
    RC5mode         = rc_7bit;                     //rc_6bit = default
```

Funktionen und Prozeduren

RecvRC5 SendRC5

1.45 RTC Driver

Import *SysTick, RTclock, ..;*

Option

From *RTclock* ***Import*** *RTCtimer, RTCalarm;*

Define

```
    ProcClock   = 6000000;          {Hertz}
    SysTick     = 10 , Timer2;      {msec}
    RTclock     = iData, Time;      {Time or DateTime}
```

Options

```
    RTCtimer    = 4;                {1..8 Channels}
    RTCsource   = SysTick[, adj];   {optional}
```

Funktionen und Prozeduren

RTCalarm	RTCgetHour	RTCsetDay	RTCsetYear	RTCtimer_Start
RTCalarm_Date	RTCgetMinute	RTCsetHour	RTCtickHour	RTCtimer_Stop
RTCalarm_Start	RTCgetMonth	RTCsetMinute	RTCtickMinute	
RTCalarm_Stop	RTCgetSecond	RTCsetMonth	RTCtickSecond	
RTCalarm_Time	RTCgetWeekDay	RTCsetSecond	RTCtimer	
RTCgetDay	RTCgetYear	RTCsetWeekDay	RTCtimer_Load	

1.46 Serial LAN

Import *LANport;*

Define

```
    LANport     = SerPort;      {SerPort2}
    LANctrl     = PortA, 5;     {PortName, bit nummer} optional
    LANmode     = Master;       {Master/Slave}
    LANbaud     = 57600;        {Baudrate}
    LANadr      = 8 [,Mask];    {8 or 16 bits, [masked by LANADRMASK]
    LANframe    = 16, iData;    {Framesize max. 16 bytes in iData}
    LANcheck    = ChkSum8;      {ChkSum8, ChkSum16, CRC16}
```

```
Var
  LANrxBuff : array [0..LANframe-1] of byte;
  LANtxBuff : array [0..LANframe-1] of byte;
```

Funktionen und Prozeduren

LANrxAutoAck	LANrxStat	LANtxFrame
LANrxClear	LANtxClear	LANtxStat

1.47 SerPort

Amerkung:
die früheren Bezeichner für den 1. SerPort, wie SerPort, SerPortDTR, SerInp etc.
wurden durch die Ziffer "1" ergänzt (umbenannt) : SerPort1, SerPortDTR1, SerInp1 etc.
Die alten Namen stehen jedoch weiterhin zur Verfügung.

```
Import SysTick, SerPort1, SerPort2, ..;
XMega:
Import SysTick, SerPortC0, SerPortC1, ..;
```

Option

```
From SerPort import SerPortSelect;
```

```
Define
  ProcClock     = 4000000;              {4Mhz clock }
  SysTick       = 10;                   {10msec Tick}
  SerPort1      = 9600, Stop1;          {9600 Baud, 1Stopbit}
  //SerPort1    = 9600, parEven;        {9600 Baud, gerade parity}
  //SerPort1    = 9600, parOdd;         {9600 Baud, odd parity}
  //SerPort1    = 9600, Stop2, timeout; // Stop2 and timeout are optional
  TxBuffer1     = 8;                    {8 Byte Buffer and Int}
  RxBuffer1     = 8, iData;             {8 Byte Buffer and Int}
```

Für die weiteren seriellen Schnittstellen lauten die Definitionen analog:

```
  SerPort2      = 9600, Stop1;          {9600 Baud, 1Stopbit}
. . .
```

XMega

```
Define
  ProcClock     = 4000000;              {4Mhz clock }
  SysTick       = 10;                   {10msec Tick}
  SerPortC0     = 9600, Stop1;          {9600 Baud, 1Stopbit}
  //SerPortC0   = 9600, parEven;        {9600 Baud, gerade Parität}
  //SerPorC0    = 9600, parOdd;         {9600 Baud, ungerade Parität}
  TxBufferC0    = 8;                    {8 Byte Buffer und Int}
  RxBufferC0    = 8, iData;             {8 Byte Buffer und Int}
```

Für die weiteren seriellen Schnittstellen lauten die Definitionen analog:

```
  SerPortC1     = 9600, Stop1;          {9600 Baud, 1Stopbit}
  . . .
```

Options

```
  SerPortDTR1 = PinB, 2, Positive;      // first serport
  SerPortDTR2 = PinB, 3, Negative;      // second serport if available
  . . .
```

```
SerPortDTRC0 = PortC, 2, Positive;    // XMega first serport
SerPortDTRC1 = PinC, 3, Negative;     // XMega second serport if available
. . .

SerPortDSR1 = PortC, 2, Positive;     // first serport
SerPortDSR2 = PortC, 3, Negative;     // second serport if available
. . .
SerPortDSRC0 = PortC, 2, Positive;    // XMega first serport
SerPortDSRC1 = PortC, 3, Negative;    // XMega second serport if available
. . .

SerCtrl1      = PortD, 2, positive;   { control line for RS485 driver }
SerCtrl2      = PortD, 3, positive;   { control line for RS485 driver }
. . .
SerCtrlC0     = PortD, 2, positive;   {XMega control line for RS485 driver}
//SerCtrlC1   = PortD, 2, positive;   {XMega control line for RS485 driver}
```

SerPortSelect

```
SerPortSelect := 3; // switch to serport4
SerPortSelect := 2; // switch to serport3
SerPortSelect := 1; // switch to serport2
SerPortSelect := 0; // switch to serport1
```

XMega

Hier ist die Variable *SerPortSelect* kein Byte sondern vom Typ tUSARTenum

```
SerPortSelect := UsartC1;             // switch to serportC1
SerOut ('x');                         // write 'x' to serportC1
if SerStat then                       // check serportC1
  ch:= SerInp;                  // read from SerportC1
endif;

SerPortSelect := UsartD0;             // switch to serportD0
SerOut ('x');                         // write 'x' to serportD0
. . .
```

Type

```
tParity = (parNone, parEven, parOdd);
tDataBits = (DataBit5, DataBit6, DataBit7, DataBit8);
tStopBits = (StopBit1, StopBit2);
```

XMega:

```
tUSARTenum = (UsartC0, UsartC1, UsartD0, UsartD1, UsartE0, UsartE1, UsartF0, UsartF1);
```

Funktionen und Prozeduren ATMega Port1

FlushBuffer	SerInp1	SerInp_TO1	SerPort_Send1
OnSerRxResumed1	SerInpBlock1	SerOut1	SerStat1
OnSerRxStopped1	SerInpBlock1_P	SerOutBlock1	SerStopBits1
SerBaud1	SerInpBlockP_TO1	SerOutBlock1_P	
SerDataBits1	SerInpBlock_TO1	SerOutSLIP1	
Ser_Enable1	SerInpSLIP1	SerParity1	

Funktionen und Prozeduren ATMega Port2

FlushBuffer	SerInp2	SerInp_TO2	SerStopBits2
OnSerRxResumed2	SerInpBlock2	SerOut2	SerPort_Send2
OnSerRxStopped2	SerInpBlock2_P	SerOutBlock2	SerStat2
SerBaud2	SerInpBlockP_TO2	SerOutBlock2_P	
SerDataBits2	SerInpBlock_TO2	SerOutSLIP2	
Ser_Enable2	SerInpSLIP2	SerParity2	

Funktionen und Prozeduren ATMega Port3, Port4: analog Port1 und Port2 (siehe oben)

Funktionen und Prozeduren XMega PortC0

FlushBuffer	SerInpC0	SerInp_TOC0	SerPort_SendC0
OnSerRxResumedC0	SerInpBlockC0	SerOutC0	SerStatC0
OnSerRxStoppedC0	SerInpBlockC0_P	SerOutBlockC0	SerStopBitsC0
SetSerBaud(UsartC0,	SerInpBlockP_TOC0	SerOutBlockC0_P	
SerDataBitsC0	SerInpBlock_TOC0	SerOutSLIPC0	
Ser_EnableC0	SerInpSLIPC0	SerParityC0	

Funktionen und Prozeduren XMega PortC1

FlushBuffer	SerInpC1	SerInp_TOC1	SerPort_SendC1
OnSerRxResumedC1	SerInpBlockC1	SerOutC1	SerStatC1
OnSerRxStoppedC1	SerInpBlockC1_P	SerOutBlockC1	SerStopBitsC1
SetSerBaud(UsartC1,	SerInpBlockP_TOC1	SerOutBlockC1_P	
SerDataBitsC1	SerInpBlock_TOC1	SerOutSLIPC1	
Ser_EnableC1	SerInpSLIPC1	SerParityC1	

Funktionen und Prozeduren XMega PortD0, PortD1, PortE0, PortE1, PortF0, PortF1: analog PortC0 und PortC1 (siehe oben)

1.48 Servo Driver

Import *ServoPort;*

Define

```
  ProcClock     = 8000000;          {8Mhz clock }
  ServoPort     = PortB, 2, iData;  {Port, Startbit im Port, Datenbereich}
  ServoChans    = 4, Positive;      (4 channels, positive pulse}
  ServoNeutral  = 1.5, Timer1;      (1.5msec neutral position, use Timer1}
  ServoSwing    = 1.5;              (adjustable between 0.5 and 1.0msec, Resolution 100
pts}
```

Alternative Auflösung von 1000 Punkten ***(*4*)***

```
  ServoSwing    = 1.5, 1000;        (adjustable between 0.5 and 1.0msec, Resolution 1000
pts}
```

Funktionen und Prozeduren

SetServoChan SetServoOffs

1.49 SHT11 Driver

Import *SysTick, SHT11drvr;*

Define

```
  ProcClock     = 8000000;          // 8Mhz clock
  SysTick       = 5;                // 5msec
  SHT11drvr     = polled [, Delay][, crc];  // polling only
  // SHT11drvr  = SysTickChecked;   // polling + semaphore
  SHT11clk      = PortD, 5;         // port and pin for clock
  SHT11dat      = PortD, 6;         // port and pin for data
```

Var

```
  SHT11sema  : Semaphore;
```

SHT11crc : byte;

Funktionen und Prozeduren

SHT11ConvState	SHT11getTemp	SHT11startTemp
SHT11getHum	SHT11setStatus	SHT11softReset
SHT11getStatus	SHT11startHum	SHT11synchronize

1.50 SLIPport

Import *SysTick…, SLIPport1, ..; // SLIPport2, SLIPport3, SLIPport4*
XMega
Import *SysTick…, SLIPportC0, ...; // SLIPporC1, SLIPportD0, SLIPportD1, SLIPportE0, SLIPportE1...*

Define
```
ProcClock       = 16000000;     {Hertz}
SysTick         = 10;
StackSize       = $0020, iData;
FrameSize       = $0040, iData;
SLIPport1       = 19200;        // SLIPport2, SLIPport3, SLIPport4
// SLIPportCtrl1 = PortA, 0, positive;   // optional RS484 line driver control
…
```
XMega
Define
```
ProcClock       = 16000000;     {Hertz}
SysTick         = 10;
StackSize       = $0020, iData;
FrameSize       = $0040, iData;
SLIPportC0      = 19200;        // SLIPportC1, SLIPportD0, SLIPportD1...
// SLIPportCtrlC0 = PortA, 0, positive; // optional RS484 line driver control
…
```

Type
```
tSLIPstate = (SLIPidle, SLIPready, SLIPbusy, SLIPovr, SLIPtout, SLIPfrm, SLIPchkE);
tSLIPmodeEn = (slpHsk, slpChkS, slpAddr);
tSLIPmode = Bitset of tSLIPmodeEn;
```

Var
```
SlipRxSema1 : semaphore;
SlipRxSema2 : semaphore;
...
```

XMega
```
SlipRxSemaC0 : semaphore;
SlipRxSemaC1 : semaphore;
SlipRxSemaD0 : semaphore;
...
```

Funktionen und Prozeduren ATMega Port1

SLIPgetRxCount1	SLIPrxReady1	SLIPsetTimeOut1	SLIPstartTxC1
SLIPgetRxState1	SLIPsetMode1	SLIPsetTxAddr1	SLIPstopRx1
SLIPgetTxState1	SLIPsetRxAddr1	SLIPsetTxBuffer1	SLIPwasBC1
SLIPresumeRx1	SLIPsetRxBuffer1	SLIPstartTx1	

Funktionen und Prozeduren ATMega Port2

SLIPgetRxCount2 SLIPrxReady2 SLIPsetTimeOut2 SLIPstartTxC2
SLIPgetRxState2 SLIPsetMode2 SLIPsetTxAddr2 SLIPstopRx2
SLIPgetTxState2 SLIPsetRxAddr2 SLIPsetTxBuffer2 SLIPwasBC2
SLIPresumeRx2 SLIPsetRxBuffer2 SLIPstartTx2

Funktionen und Prozeduren ATMega Port3, Port4: analog zu Port1 und Port2 (siehe oben)

Funktionen und Prozeduren XMega PortC0

SLIPgetRxCountC0 SLIPrxReadyC0 SLIPsetTimeOutC0 SLIPstartTxC_C0
SLIPgetRxStateC0 SLIPsetModeC0 SLIPsetTxAddrC0 SLIPstopRxC0
SLIPgetTxStateC0 SLIPsetRxAddrC0 SLIPsetTxBufferC0 SLIPwasBC_C0
SLIPresumeRxC0 SLIPsetRxBufferC0 SLIPstartTxC0

Funktionen und Prozeduren XMega PortC1

SLIPgetRxCountC1 SLIPrxReadyC1 SLIPsetTimeOutC1 SLIPstartTxC_C1
SLIPgetRxStateC1 SLIPsetModeC1 SLIPsetTxAddrC1 SLIPstopRxC1
SLIPgetTxStateC1 SLIPsetRxAddrC1 SLIPsetTxBufferC1 SLIPwasBC_C1
SLIPresumeRxC1 SLIPsetRxBufferC1 SLIPstartTxC1

Funktionen und Prozeduren XMega PortD0, PortD1, PortE0, PortE1, PortF0, PortF1: analog zu PortC0 und PortC1 (siehe oben)

1.51 Software PWM

Import *SysTick, SoftPWM;*

Define

```
ProcClock       = 8000000;      {Hertz}
SysTick         = 10;           {msec}
StackSize       = $0032, iData;
FrameSize       = $0032, iData;
SoftPWMport     = PortA;        {use PortA for PWM output}
SoftPWMchans    = 4, 0;         {4 channels, bit0/PortA is the first}
//SoftPWMchans  = 4, 2, negative; {4 channels, bit2/PortA 1.bit, low pulsed}
SoftPWMtimer    = timer2, 1;    {use timer2, PWM cycle time 10msec = 200Hz}
SoftPWMres      = 16;           {PWM resolution is 16 points}
```

Var

SoftPWM1, SoftPWM2, ... : Byte;

Funktionen und Prozeduren

SoftPWMstart SoftPWMstop

1.52 Speech Port

Import *SpeechPort ;* *{not SPI and **XMega**}*
oder
Import *SpeechPort, SPIdriver ;*

Define

```
ProcClock    = 8000000;    {8Mhz clock }
SpeechPort   = SPI;        // SPIdriver must be imported and defined
```

oder

```
  SpeechPort    = UserPort;      // uses an user defined driver
oder
  SpeechPort    = PortG, 0;      // bit serial using PortG, bit0=DATA, 1=CLK, 2=SEL
oder
  SpeechPort    = PortC;         // 8bit parallel output on port C
  SpeechTimer   = Timer0;        // Timer0, Timer1, Timer2, Timer3
oder
  SpeechTimer   = Timer1;
oder
  SpeechTimer   = Timer2;
oder
  SpeechTimer   = Timer3;
  SPIorder      = LSB;           // SPI define only necessary if SpeechPort = SPI;
  SPIcpol       = 1;
  SPIcpha       = 1;
  SPIpresc      = 0;             // presc = 0..3 -> 4/16/64/128
  SPI_SS        = true;          // use SS pin as chipselect
...
oder XMega
  SpeechTimer   = Timer_C0;      //_C1, _D0, _D1, ...
  SpeechPort = SPI_C, SPImode3, SPImsb, PortF, 4; // Mode 0..3, MSB/LSB, SS-Port, SS-Pin

UserDevice SpeechIOS(b : byte);
begin
  ...
end;
```

Funktionen und Prozeduren

SpeechOutFlash SpeechOutRAM SpeechReady SpeechStop

1.53 SPI Driver MSPI-ATMega

```
Import SysTick, MSPI0, MSPI1, ..;

Define
  ProcClock     = 8000000;       {Hertz}
  SysTick       = 10;            {msec}
  StackSize     = $0030, iData;
  FrameSize     = $0030, iData;
  MSPIorder0    = MSB;
  MSPIcpol0     = 1;
  MSPIcpha0     = 1;
  MSPIpresc0    = 1;
   . . . .
  MSPIpresc1    = 1;
```

Funktionen und Prozeduren für Port MSPI0

MSPIinOut0	MSPIinpByte0	MSPIout0	MSPIoutWord0	SetMSPIclkPol0
MSPIinOutByte0	MSPIinpLong0	MSPIoutByte0	SetMSPI0mode	SetMSPIorder0
MSPIinp0	MSPIinpWord0	MSPIoutLong0	SetMSPIclkPha0	SetMSPIpresc0

Funktionen und Prozeduren für Port MSPI1

MSPIinOut1	MSPIinpByte1	MSPIout1	MSPIoutWord1	SetMSPIclkPol1
MSPIinOutByte1	MSPIinpLong1	MSPIoutByte1	SetMSPI1mode	SetMSPIorder1
MSPIinp1	MSPIinpWord1	MSPIoutLong1	SetMSPIclkPha1	SetMSPIpresc1

Funktionen und Prozeduren für Port MSPI2 und MSPI3: analog MSPI0 und MSPI1(siehe oben)

1.54 SPI Driver MSPI-XMega

Import *SysTick, MSPI_C0, MSPI_C1, ..;*

Define

```
// The XMegas don't provide any Oscillator fuses.
// So the application must setup the desired values
// possible OSC types: extXTAL, extClock, ext32kHz, int32Khz, int2MHz, int32MHz
//>> CPU=32MHz, PeripherX4=32MHz, PeripherX2=32MHz
OSCtype             =   int32MHz,
                        PLLmul=4,
                        prescB=1,
                        prescC=1;
SysTick             =   10;             {msec}
StackSize           =   $0030, iData;
FrameSize           =   $0030, iData;
MSPIorder_C0        =   MSB;
MSPIMode_C0         =   1;              // 0, 1, 2, 3
MSPIpresc_C0        =   1;              // presc = 1..7
MSPI_SS_C0          =   PortE, 4;
MSPIorder_C1        =   MSB;
MSPImode_C1         =   0;              // 0, 1, 2, 3
MSPIpresc_C1        =   3;              // presc = 1..7
MSPI_SS_C1          =   none;
....
```

Funktionen und Prozeduren für Port MSPI_C0

MSPIinOut_C0	MSPIinpLong_C0	MSPIoutLong_C0	SetMSPImode_C0
MSPIinOutByte_C0	MSPIinpWord_C0	MSPIoutWord_C0	SetMSPIorder_C0
MSPIinp_C0	MSPIout_C0	SetMSPIclkPha_C0	SetMSPIpresc_C0
MSPIinpByte_C0	MSPIoutByte_C0	SetMSPIclkPol_C0	

Funktionen und Prozeduren für Port MSPI_C1

MSPIinOut_C1	MSPIinpLong_C1	MSPIoutLong_C1	SetMSPImode_C1
MSPIinOutByte_C1	MSPIinpWord_C1	MSPIoutWord_C1	SetMSPIorder_C1
MSPIinp_C1	MSPIout_C1	SetMSPIclkPha_C1	SetMSPIpresc_C1
MSPIinpByte_C1	MSPIoutByte_C1	SetMSPIclkPol_C1	

Funktionen und Prozeduren für Port MSPI_D0, D1, E0, E1, F0, F1: analog Port MSPI_C0 und MSPI_C01 (siehe oben)

1.55 SPI Hardware Driver

Import *SysTick, SPIdriver, ..;*

XMega
Import *SysTick, SPI_C, ..; // SPI_D, SPI_E, SPI_F*

Define
```
ProcClock   = 8000000;      {Hertz}
SysTick     = 10;           {msec}
StackSize   = $0030, iData;
FrameSize   = $0030, iData;
SPIorder    = MSB;
SPIcpol     = 1;
SPIcpha     = 1;
SPIpresc    = 1;            // presc = 0..3 -> 4/16/64/128
SPI_SS      = false;        // don't use SS pin as chipselect
```

XMega
Define
```
OSCtype     = int32MHz, PLLmul=4, prescB=1, prescC=1; // all 32MHz
SysTick     = 10;           {msec}
StackSize   = $0030, iData;
FrameSize   = $0030, iData;
SPIorderC   = MSB;
SPImodeC    = 0;            // Clock Phase and Polarity
SPIprescC   = 1;            // presc = 0..3 -> 4/16/64/128
SPI_SSC     = PortB, 3;     // use this pin as SS chipselect
```

Funktionen und Prozeduren ATMega

SetSPIclkPha	SetSPIorder	SPIinOutByte	SPIout
SetSPIclkPol	SetSPIpresc	SPIinpByte	SPIoutByte
SetSPIdoubleSpeed	SPIinp	SPIinpLong	SPIoutLong
SetSPImode	SPIinOut	SPIinpWord	SPIoutWord

Funktionen und Prozeduren XMega Port_C

SetSPIclkPhaC	SetSPIprescC	SPIinpLongC	SPIoutLongC
SetSPIclkPolC	SPIinpC	SPIinpLong64C	SPIoutLong64C
SetSPIdoubleSpeedC	SPIinOutC	SPIinpWordC	SPIoutWordC
SetSPImodeC	SPIinOutByteC	SPIoutC	
SetSPIorderC	SPIinpByteC	SPIoutByteC	

Funktionen und Prozeduren XMega Port_D

SetSPIclkPhaD	SetSPIprescD	SPIinpLongD	SPIoutLongD
SetSPIclkPolD	SPIinpD	SPIinpLong64D	SPIoutLong64D
SetSPIdoubleSpeedD	SPIinOutD	SPIinpWordD	SPIoutWordD
SetSPImodeD	SPIinOutByteD	SPIoutD	
SetSPIorderD	SPIinpByteD	SPIoutByteD	

Funktionen und Prozeduren XMega Port_E und Port_F:
analog Port_C und Port_D (siehe oben)

1.56 SPI Network

```
Import SysTick, SPIport;

Define
   SPIport      = Master;
or
   SPIport      = Slave;
   SPIOrder     = MSB; {LSB}
   SPICPOL      = 0;          {0/1}
   SPICPHA      = 0;          {0/1}
   SPIpresc     = 0;          {0/1/2/3}
   SPIretry     = 10;

Var
   SpiTxBuff : array [0..SPIbufferLen -1] of byte;
   SpiRxBuff : array [0..SPIbufferLen -1] of byte;
   SpiRxLen : byte;
   SpiTxLen : byte;
```

Funktionen und Prozeduren

SPIinp	SPIrxClear	SPIrxStat	SPItxFrame
SPIout	SPIrxFrame	SPItxClear	SPItxStat

1.57 SPI Soft Driver

```
Import SysTick, SPIdriver1, ..;          // or SPIdriver2 or both

Define
   ProcClock    = 16000000;          {Hertz}
   SysTick      = 10;                {msec}
   StackSize    = $0030, iData;
   FrameSize    = $0040, iData;
   SPIdriver1   = PortA, 0, 1, 2, 3;     // SCK, MOSI, MISO, SS
   SPIorder1    = MSB;
   SPIcpol1     = 1;
   SPIcpha1     = 1;
   SPIdriver2   = PortC, 7, 4, 1, 5;     // SCK, MOSI, MISO, SS
   SPIorder2    = MSB;
   SPIcpol2     = 0;
   SPIcpha2     = 0;
```

Funktionen und Prozeduren

SPIinOutByte1	SPIinpByte1	SPIinpWord1	SPIoutByte1	SPIoutWord1
SPIinOutByte2	SPIinpByte2	SPIinpWord2	SPIoutByte2	SPIoutWord2
SPIinOut1	SPIinpLong1	SPIinp1	SPIoutLong1	SPIout1
SPIinOut2	SPIinpLong2	SPIinp2	SPIoutLong2	SPIout2

1.58 Stepper Driver

Import *SysTick, StepPort..*

From *System* ***Import*** *longword;*

Option
From *StepPort* ***Import*** *StepperSema;*

Define
ProcClock = 8000000; {Hertz}
StackSize = $0030, iData;
FrameSize = $0010, iData;
StepPort = PortA;
StepMinFreq = 100;
StepMaxFreq = 5000;
StepType = StepM6;
XMega
StepTimer = Timer_C0; {Timer _C1, _D0, _D1, ...}

StepTypes
StepFull 2, StepFull 4, SteppHalf 4, StepHalf 6, StepMini 4
StepMini 6, StepMicro 2, StepMicro 8, UserPort

Type
TStepMode = (StepStop, StepUp, StepRun, StepDown);

Var
StepStartFreq : word;
StepEndFreq : word;
StepAccValue : word;
StepCount : longword; {max 2^32 Steps}
StepMode : TStepMode;

Funktionen und Prozeduren

StepDestCCW	StepperOff	StepPanicStop	StepRampStop
StepDestCW	StepOneCW	StepRampCW	StepVelocity
StepperOn	StepOneCCW	StepRampCCW	

1.59 Switchport

Import *SysTick, SwitchPort1, ..;*
or
Import *SysTick, SwitchPort2, ..;*
or
Import *SysTick, SwitchPort_G, ..;*

Define
ProcClock = 8000000; {Hertz}
SysTick = 10; {msec}
StackSize = $0030, iData;
FrameSize = $0030, iData;
SwitchPort1 = PinB; {SwitchPort}
or

```
  SwitchPort1   = PinB, $17;      {SwitchPort, edgemask}
  SwitchPort_G = [ResetKey, PinC, 4] [StartInput, PinC, 5] [Sensor, PinB, 1], %00000001;
options
  PolarityP1    = %00000110;  {Polarity SwitchPort1}
  Debounce      = 5;          {debounce every 5 SysTicks}

Var
  PORT_STABLE1      : byte;
  PORT_STABLE2      : byte;
  PORT_STABLE_G     : byte;
```

Funktionen und Prozeduren

Inp_Raise1	Inp_Stable1	SwitchKeyRepeat1	SwitchPort1_Clear
Inp_Raise2	Inp_Stable2	SwitchKeyRepeat2	SwitchPort2_Clear
Inp_Raise_G	Inp_Stable_G	SwitchKeyRepeat_G	SwitchPort_G_Clear

1.60 SysLeds

```
Import SysTick, SysLEDblink, ..;
```

Options

```
From SysLEDblink Import LEDmessage, FlashOnce;     // this is optional

Define
  ProcClock     = 8000000;              {Hertz}
  SysTick       = 10;                   {msec}
  StackSize     = $0030, iData;
  FrameSize     = $0030, iData;
  SysLEDblink   = 30;                   {30*SysTick = 300msec}
  // alternative
  //SysLEDblink = mSec300;              {10..1000 msec}
  SysLEDBlink0 = PortA, 0, high;        {LEDon = high level}
  SysLEDBlink1 = PortA, 1, low;         {LEDon = low level}
  SysLEDBlink2 = PortA, 2, low;
  SysLEDBlink3 = PortA, 3, low;
  SysLEDBlink4 = PortA, 4, low;
  SysLEDBlink5 = PortA, 5, low;
  SysLEDBlink6 = PortA, 6, low;
  SysLEDBlink7 = PortA, 7, low;
  // alternative
 //SysLedPort = @LEDram, $00;           {byte-var, polarity}
```

Funktionen und Prozeduren

SetSysBlinkTimer	SysLEDflashAllOn	SysLEDflashOn	SysLEDon
SysLEDallOn	SysLEDflashAllOff	SysLEDflashOnce	SysLEDonOff
SysLEDallOff	SysLEDflashMsg	SysLEDflashOnOff	
SysLEDenable	SysLEDflashOff	SysLEDoff	

1.61 Tick Timer, TickTimer2

Import *SysTick, TickTimer, ..;*

Define

```
  ProcClock     = 16000000;          {Hertz}
  SysTick       = 10;                {msec}
  StackSize     = $0030, iData;
  FrameSize     = $0030, iData;
  TickTimer     = Timer1;            // use Timer1.COMPA and no PortPin
  //TickTimer   = Timer1, pinout;    // use Timer1.COMPA and its PortPin
  //TickTimer   = Timer2;            // use Timer2.COMPA and no PortPin
  //TickTimer   = Timer2, pinout;    // use Timer2.COMPA and its PortPin
  //TickTimer   = Timer3;            // use Timer3.COMPA and no PortPin
  //TickTimer   = Timer3, pinout;    // use Timer3.COMPA and its PortPin
```

Var *TickTimerPin : bit;*

xMega:
unterstützt zusätzlich TickTimer2

Define

```
 TickTimer     = Timer_C0;
 TickTimer2    = Timer_C1;
oder
 TickTimer     = Timer_D0, PortE, 4;
 TickTimer2    = Timer_D1, PortE, 5;
```

Funktionen und Prozeduren

TickTimerOutpEnable	TickTimerReload	TickTimerStop
TickTimerRawVal	TickTimerStart	TickTimerTime

zuätzliche Funktionen und Prozeduren für xMega

TickTimer2OutpEnable	TickTimer2Reload	TickTimer2Stop
TickTimer2RawVal	TickTimer2Start	TickTimer2Time

1.62 TINA TCP/IP

Import *SysTick, TINAstack, ...;*

Define

```
 ProcClock = 16000000;          // Hertz
 StackSize = $0080, iData;      // min size
 FrameSize = $00C0, iData;      // min size
 xData = $8000, $87ff;          // 2kB optional, only necessary with the xData define below
 TINAdriver = ENC28J60[, xData]; // TINA hardware,optional buffers in xData
 // TINAdriver = ENC424J600;    // TINA hardware, alternative chip
 TINAport = SPI, PortB, 0;      // SPItyp, SS_Port, SS_Pin
 // TINAport = MSPI0, PortA, 4; // SPItyp, SS_Port, SS_Pin
 TINAtimer = Timer3;            // 1..3
 TINAsockets = 4;               // socket count, 1..8
```

XMega
XMegas support up to four SPI-ports:

SPI_C, SPI_D, SPI_E or SPI_F

The SPI Port must be defined:

TINAport = SPI_C, PortF, Pin3;

XMegas have up to 8 Timers:

Timer_C0, Timer_C1, Timer_D0, Timer_D1 etc.

One Timer must be used:

TINAtimer = Timer_F1;

```
Type
TMACaddr        = array [0..5] of byte;
TIPaddr         = array [0..3] of byte;

TTINAStatus =   (TinasNoErrors, TinasInvalidHandle, TinasInitFailed,
                 TinasNotInitialized, TinasSockClosed, TinasBufferParam,
                 TinasSendFailed, TinasTimeOutErr, TinasListenFailed,
                 TinasSockConnected, TinasSockListen, TinasSockCloseWait,
                 TinasSockClosing, TinasSockUDP, TinasSockRaw);

TTinaPriority =  (TinaPrioMedium, TinaPrioLow, TinaPrioHigh, TinaPrioVeryHigh,
                 TinaPrioVeryLow, TinaPrioAuto, TinaPrioSuspend, TinaPrioResume);

TtinaPacketReceive = Record
                       PeerIP       : tIPaddr;
                       PeerPort     : Word;
                       RecLen       : Word;
                       BufferPtr    : Pointer;
                       BufferLen    : Word;
                     end;

TTinaSocketSWS      = (NoSillyWindow, SillyWindow);
TTinaSocketNDAck    = (NoDelayedAck, DelayedAck);
TTinaNDTimeOut      = (NoDynamicTimeOut, DynamicTimeOut);
TTinaBroadcast      = (NoBroadcast, Broadcast);
TTinaxUDPAKNPort    = (xAKNLocalPort, xAKNRemotePort);

TTinaSocket = Record
                Protocol            : TProtocolType;
                Swindow             : TTinaSocketSWS;
                DelayAck            : TTinaSocketNDAck;
                DynTimeOut          : TTinaNDTimeOut;
                Broadcast           : TTinaBroadcast;
                LocalPort           : Word;
                RemoteHost          : tIPaddr;
                RemotePort          : Word;
                TimeOut             : LongWord;
                RetryCount          : Byte;
                AKNPort             : TTinaxUDPAKNPort;
                PeerTryToDisconnect : Boolean;
                SocketClosed        : Boolean;
                PacketRecInfo       : TTinaPacketReceive;
                ErrorState          : TTINAStatus;
                SocketState         : Byte;
              end;
tSocketHandle = Pointer to TTinaSocket;

TTinaCore = Record
```

```
    IP                  : tIPaddr;
    Mask                : tIPaddr;
    Gateway             : tIPaddr;
    Mac                 : TMacAddr;
    TimeOut             : LongWord;
    Retry               : Byte;
    Prio                : TTinaPriority;
    ResponsePing        : Boolean;
    RXCheckSumCheck     : Boolean;
    SendICMPCtrlMessages : Boolean;
  end;

var
  TinaCore : TTinaCore;
```

Funktionen und Prozeduren

CompareIP	StrToMAC	TINAlinkStat	TINA_Init
CompareNet	SwapIPaddr	TINApacketReceived	TINA_Ping
CompareMAC	SwapMACaddr	TINAresumeReceive	TINA_Start
IPtoStr	TINAcreateSocket	TINArxStat	TINA_Stop
MACtoStr	TINAfreeSocket	TINAsendPacket	
StrToIP	TINAinitSocket	TINASetPriority	

1.63 TWI (I2C) Port

```
Import SysTick, TWImaster;

Define
  ProcClock    = 4000000;     {4Mhz clock }
  SysTick      = 10;          {10msec Tick}
  TWIpresc     = 32           (TWI speed}
```

or

```
Import SysTick, TWIslave;
```

or ***XMega***

```
Import SysTick, TWI_C;        // TWI_C, TWI_D, TWI_E, TWI_F
```

Achtung:
Bei den XMegas sind die MasterSlave und Slave Modes nicht unterstützt!

```
Define
  ProcClock    = 4000000;     {4Mhz clock }
  SysTick      = 10;          {10msec Tick}
  TWIaddr      = 2;           (TWI slave address}
  TWIbuffer    = 8, iData;    (TWI rx/tx buffersize, location}
  TWImode      = Handshake;   (TWI handshake or transparent}
```

XMega

```
Define
  OSCtype      = int32MHz, PLLmul=4, prescB=1, prescC=1;
  SysTick      = 10;          {10msec Tick}
  TWIprescC    = TWI_BR400;   (TWI speed}

Const
  TWI_BR100 : byte = nn;      // nn = prescaler value for 100kBits/sec
  TWI_BR400 : byte = nn;      // nn = prescaler value for 400kBits/sec
```

```
TWI_BR500 : byte = nn;      // nn = prescaler value for 500kBits/sec
TWI_BR600 : byte = nn;      // nn = prescaler value for 600kBits/sec
TWI_BR800 : byte = nn;      // nn = prescaler value for 800kBits/sec
```

Var

```
TWI_DevLock : DEVICELOCK;
TWI_DevLockTN : DEVICELOCK;     // TN = C, D, E, F
```

Statusbytes im TWISlave Mode ohne Handshake: *TWIslvTxCount, TWIslvRxCount.*

Funktionen und Prozeduren

TWIgetBusy	TWIgetRxStat	TWIinpP	TWIoutWP	TWIsetRdy
TWIgetCmd	TWIgetTxStat	TWIout	TWIsetBusy	TWIsetSlaveAddr
TWIgetRdy	TWIinp	TWIoutP	TWIsetGC	TWIstat

Funktionen und Prozeduren für XMega

TWIinpC	TWIinpD	TWIinpE	TWIinpF
TWIinpPC	TWIinpPD	TWIinpPE	TWIinpPF
TWIoutC	TWIoutD	TWIoutE	TWIoutF
TWIoutPC	TWIoutPD	TWIoutPE	TWIoutPF
TWIoutWPC	TWIoutWPD	TWIoutWPE	TWIoutWPF
TWIstatC	TWIstatD	TWIstatE	TWIstatF

1.64 TWI Network

Import *SysTick, TWInet;*

Define

```
ProcClock    = 16000000;    {16Mhz clock }
TWIpresc     = TWI_BR100;   {TWI speed 100kBit/sec}
TWInode      = $12;         {Node Addr, always necessary}
TWInetMode   = Master;      {Master, Slave or MasterSlave}
TWIframe     = 16, iData;   {Framesize max. 16 bytes in iData}
```

Option

```
TWIframeBC   = 64;          {Broadcast Framesize max. 16 bytes}
```

Type

```
tTWInetmode = (TWInetSlave, TWInetMaster);
tTWIStates  = (TWIRxEmpty, TWIRxBusy, TWIRxFull, TWITxEmpty,
               TWITxBusy, TWITxFull, TWIbcCMD, TWIstatFail);
tTWINetState = set of tTWIStates;
```

Const

```
TWI_BR100 : byte = nn;      // nn = prescaler value for 100kBits/sec
TWI_BR400 : byte = nn;      // nn = prescaler value for 400kBits/sec
TWI_BR500 : byte = nn;      // nn = prescaler value for 500kBits/sec
TWI_BR600 : byte = nn;      // nn = prescaler value for 600kBits/sec
TWI_BR800 : byte = nn;      // nn = prescaler value for 800kBits/sec
```

Var

```
TWIrxStatReg        : byte;
TWItxStatReg        : byte;
TWIrxAdr            : byte;
TWItxAdr            : byte;
TWIrxLen            : byte|word;
TWItxLen            : byte|word;
```

```
TWIrxBuff       : array [0..TWIframe -1] of byte;
TWItxBuff       : array [0..TWIframe -1] of byte;

TWI_DevLock : DEVICELOCK;
```

Funktionen und Prozeduren

GetTWIslaveStat	TWIrxClear	TWItxBroadcast	TWItxStat
SetTWInodeAddr	TWIrxFrame	TWItxClear	
SetTWImode	TWIrxStat	TWItxFrame	

1.65 wizNet TCP/IP

```
Import SysTick, TWIMaster, wzNet4, ..;

From System Import Tasks, Processes;

Define
  ProcClock     = 16000000;          // Hertz
  SysTick       = 5;                 // msec
  StackSize     = $0040, iData;      // min size
  FrameSize     = $00C0, iData;      // min size
  Scheduler     = iData;
  TaskStack     = $0040, iData;      // min size
  TaskFrame     = $00C0;             // min size
  wzNet4        = I2C_TWI, iData;    // hardware I2C driver, var loc
  wzSocks       = 1;                 // socket count, 1..4
  TWIpresc      = TWI_BR400;         // max TWI speed

Import SysTick, I2Cport, wzNet4, ..;

From System Import Tasks, Processes;

Define
  ProcClock     = 16000000;          // Hertz
  SysTick       = 5;                 // msec
  StackSize     = $0040, iData;      // min size
  FrameSize     = $00C0, iData;      // min size
  Scheduler     = iData;
  TaskStack     = $0040, iData;      // min size
  TaskFrame     = $00C0;             // min size
  wzNet4        = I2C_Soft, iData;   // software I2C driver, var loc
  I2Cport       = PortA;
  I2Cclk        = 1;
  I2Cdat        = 2;
  wzSocks       = 1;                 // socket count, 1..4

Type
  TMACaddr      = array [0..5] of byte;
  TIPaddr       = array [0..3] of byte;
  TwzStatus     = (wzsNoErrors, wzsInvalidHandle, wzsInitFailed,
                   wzsNotInitialized, wzsSockClosed, wzsBufferParam,
                   wzsSendFailed, wzsTimeOutErr, wzsListenFailed,
                   wzsSockConnected, wzsSockListen, wzsSockCloseWait,
                   wzsSockClosing, wzsSockUDP, wzsSockRaw);
  TwzPriority   = (wzPrioLow, wzPrioMedium, wzPrioHigh, WzPrioVeryHigh,
                   wzPrioAuto, wzPrioSuspend, wzPrioResume);
```

```
TwzPacketReceive = Record
                     PeerIP      : tIPaddr;
                     PeerPort    : Word;
                     BufferPtr   : Pointer;
                     BufferLen   : Word;
                   end;
TwzSocketSWS = (NoSillyWindow, SillyWindow);              // internal use
TwzSocketNDAck = (NoDelayedAck, DelayedAck);              // internal use
TwzNDTimeOut = (NoDynamicTimeOut, DynamicTimeOut);        // internal use
TwzBroadcast = (NoBroadcast, Broadcast);                  // internal use
TwzSocketProtocol = (CLOSED, protTCP, protUDP, protIPRAW, protMACRaw);
TwzSocket = Record
              Protocol          : TwzSocketProtocol;
              SWindow           : TwzSocketSWS;         // internal use
              DelayAck          : TwzSocketNDAck;       // internal use
              DynTimeOut        : TwzNDTimeOut;         // internal use
              Broadcast         : TwzBroadcast;         // internal use
              LocalPort         : Word;
              RemoteHost        : tIPaddr;              // Client mode
              RemotePort        : Word;                 // Client mode
              IPProtocol        : Byte;                 // internal use
              TypeOfService     : Byte;                 // internal use
              MaxSegSize        : Word;                 // internal use
              PeerTryToDisconnect : Boolean;
              SocketClosed      : Boolean;
              PacketRecInfo     : TwzPacketReceive;
              ErrorState        : TwzStatus;            // internal use
              SocketState       : byte;                 // semaphore
            end;
tSocketHandle = Pointer to twzSocket;

var
  TWI_DevLock : DEVICELOCK;
```

Funktionen und Prozeduren

CompareIP	wzCreateSocket	wzReset	wzTelnetClose
CompareNet	wzDisConnect	wzResumeReceive	wzTelnetConnected
CompareMAC	wzDNSQueryHost	wzSendBuffer	wzTelnetCreate
IPtoStr	wzFreeSocket	wzSetDNSserver	wzTelnetEcho
MACtoStr	wzGetLastError	wzSetGatewayAddr	wzTelnetFree
StrToIP	wzGetSocketState	wzSetHWAddr	wzTelnetGetClient
StrToMAC	wzInit	wzSetIPAddr	wzTelnetGetState
SwapIPaddr	wzInitSocket	wzSetPriority	wzTelnetIdleTimeout
SwapMACaddr	wzListen	wzSetRetryCount	wzTelnetListen
wzAcceptConnection	wzPacketReceived	wzSetSNTPserver	wzTelnetRead
wzClientConnected	wzReceiveBuffer	wzSetTimeOut	wzTelnetWrite
wzConnect	wzReInitSocket	wzSNTPQueryDateTime	wzTelnetWriteLn

2 Compiler Schalter

2.1 $ANALYSIS_ON

{$ANALYSIS_ON}

Group: CompilerSwitch 66

Compiler Schalter für den Optimizer.
Gibt das Erzeugen des Analyse File frei.

2.2 $BDATA

{$BDATA #}

Group: CompilerSwitch 66

Banked External Memory
Um die verschiedenen Banks mit Variablen zu besetzen muss der User angeben in welcher Bank eine bestimmte Variable angesiedelt werden soll.
Dies geschieht durch den vorangestellten Compiler Switch {$BDATA #} wobei # die gewünschte Bank Nummer bezeichnet.

In der vorliegenden Implementation können bis zu 16 banks mit jeweils 32kB definiert und verwaltet werden.
Dies ergibt eine zusätzliche Speichergrösse von bis zu 512kBytes.

2.3 $BootApplication

*{$BootApplication $nnnn}* // word address, z.B. {$BootApplication $0F000}

Group: CompilerSwitch 66

Kombination aus $CodeStart und $VectTab. Nur für ganz spezielle Aufgaben!

Damit ist es z.B. möglich eine Applikation zu erstellen, die nur im Boot Bereich angesiedelt ist. Dies bedeutet dass das Boot/Downloader ein eigenes und von der Applikation unabhängiges Programm ist. Beide Programme, Boot und Applikation, werden vollkommen getrennt erstellt.

2.4 $BOOTRST

{$BOOTRST nnnn}

Group: CompilerSwitch 66

Dieser Schalter dient nur zur Information des Debuggers/Simulators E-LAB AVRsim, ob der Hardware Reset mit den Vektor Addr 0000 ausgeführt wird
oder in den Bootblock geht. Der Parameter $nnnn bestimmt bei aktivem Schalter die Einsprung Adresse für den Simulator bei einem Reset.
Mit der realen CPU wird mit dem Fusebit *BOOTRST* während des Programmierens durch den InCircuit Programmer bestimmt,

wie die CPU sich bei einem solchen Reset verhalten soll. Der Schalter hat also keinerlei Auswirkungen auf das generierte Programm und damit auch nicht auf das Verhalten der CPU selbst.
Der Adress Parameter kann beliebig gewählt werden, hat aber nur einen Sinn, wenn er mit den möglichen Einsprung Adressen der jeweiligen CPU übereinstimmt.

2.5 $CodeStart

*{$CodeStart $nnnn}* // word address, z.B. {$CodeStart $0F000}

Group: CompilerSwitch 66

Legt den Anfang des Code Bereichs fest. Nur für ganz spezielle Aufgaben,
wie das Erstellen einer Applikation, die direkt im BootBlock platziert wird.
siehe auch: $BootApplication und $VectTab

2.6 $D

{$D+} / {$D-}

Group: CompilerSwitch 66

Debug Informationen ein bzw. aus. Wenn aus, werden die folgenden Statements nicht im Single Step Modus abgearbeitet.
Dieser Schalter hat keinerlei Einfluss auf das generierte Hexfile, d.h. er muss nicht entfernt werden.

2.7 $DATA

{$DATA}

Group: CompilerSwitch 66

$DATA weist allen nachfolgenden Variablen Deklarationen (dazu zählt auch **StructConst**) den im Prozessor Steuerfile (xxx.dsc) unter DATA
ausgewiesenen Bereich zu. Beim AVR ist dieser Bereich von $04 bis $1F. Die definierten Variablen werden jetzt fortschreitend ab $04 platziert.
Folgt ein anderer Schalter dieses Typs, wird mit den folgenden Variablen analog dazu verfahren.
Variablen im Bereich $DATA sind immer mit
sehr kurzen und schnellen Maschinenbefehlen zu erreichen.

2.8 $DEBDELAY

{$DEBDELAY}

Group: CompilerSwitch 66

Speziell für den Simulator. Verkürzt die mDelays innerhalb des Simulators um ca. 90%. Dieser Schalter hat keinerlei Einfluss
auf das generierte Hexfile, d.h. er muss nicht entfernt werden.

2.9 $DEFINE

{$DEFINE label}

Group: CompilerSwitch 66

Setzt "label" auf true.

2.10 $DEPHASE

{$DEPHASE}

Group: CompilerSwitch 66

Schaltet auf feste **WORD**-Adresse im Flash um / zurück auf die Standard Code Page

{$PHASE $1E00}; legt den nachfolgenden Code ab Adr $1E00 ab.
{$DEPHASE} ; schaltet wieder um auf die Standard Code page.

2.11 $DEVICE

{$DEVICE}

Group: CompilerSwitch 66

Gilt nur in Zusammenhang mit Device Treiber Prozeduren, die durch die Prozedur **Write** und **Read** aufgerufen werden. Bei der nachfolgenden Device Prozedur, die nur einen 8bit Übergabe Parameter besitzen darf, wird dieser Parameter in einem Register übergeben.
Ein Parameterframe wird nicht gebildet. Damit sind auch keine lokale Variablen möglich.

Nur für schnelle in Assembler geschriebene Treiber Routinen sinnvoll. Identisch mit dem $NOFRAME Schalter.

2.12 $EEPROM

{$EEPROM}

Group: CompilerSwitch 66

$EEPROM weist allen nachfolgenden Variablen Deklarationen (dazu zählt auch **StructConst**) den im Prozessor Steuerfile (xxx.dsc) unter EEprom ausgewiesenen Bereich zu.
EEprom kann nur ein Chip-interner Speicher sein.

2.13 $EEPROM1

{$EEPROM1} **(*4*)**

Group: CompilerSwitch 66

Teilt das interne EEprom in zwei Teile.

```
Define EEprom1     = $800;
```

2.14 $ELSE

{$ELSE}

Group: CompilerSwitch 66

Kehrt den momentanen Status um. Wurde z.B. die vorhergehende Source compiliert, so wird die nachfolgende Source
bis zum "ENDIF" als Kommentar behandelt.

2.15 $ELSIF

{$ELSIF label}

Group: OverviewSwitches 66

Hiermit können einfache bool'sche Ausdrücke angegeben werden. Als Operatoren sind nur "AND" und "OR" zulässig
und als Argumente nur solche die mit {$DEFINE ..} deklariert wurden.

{$IF ABC AND XYZ}
...
{$ELSIF HIJK OR OPQ}
...
{$ENDIF}

2.16 $ELSIFDEF

{$ELSIFDEF label}

Group: CompilerSwitch 66

Wenn "true" ist, wird die nachfolgende Source compiliert.

2.17 $ENDIF

{$ENDIF}
{$IFEND}

Group: CompilerSwitch 66

Schliesst einen Conditional-Block ab.

2.18 $ENUMTOASM

{$ENUMTOASM}

Group: CompilerSwitch 66

Enumerationen (Aufzählungstypen) werden normalerweise nicht in das Assemblerfile exportiert, um Rechenzeit im Compiler
und Assembler zu sparen und die Dateien übersichtlich zu halten.

Wird im Assembler bzw. mit InLine Assembler Code die Werte der Enumeration gebraucht, so kann mit diesem Compilerschalter
der Export der Enum-Werte in das Assembler File erzwungen werden.

2.19 $HEXNAME

{$HEXNAME 'filename'}

Group: CompilerSwitch 66

Werden mit "conditional compile" aus einer Source unterschiedliche Firmware Versionen erstellt, ist es sinnvoll, die generierten Hex-Files auch unterschiedlich zu benennen. Dazu wurde dieser Compiler Schalter implementiert.
Alle beteiligten Tools, die aus der IDE heraus aufgerufen werden (Editor, Compiler, Assembler, Programmer) beachten diesen Schalter.
Das Argument 'filename' muss in Hochkomma dargestellt sein. Die beiden Schalter HexPath und HexName können auch in Kombination verwendet werden.

Bemerkung:
Eine Kopie der Flash und EEprom Hexfiles wird auch weiterhin in der Projekt Directory unter dem jeweiligen Original Namen abgelegt. Der InCircuit Programmer kann dann direkt mit dem neuen bzw.geänderten Projekt aus der IDE heraus gestartet werden. Im stand alone Betrieb des Programmers müssen dann allerdings die neuen Directories bzw. Filenamen auch als neue Projekte angelegt werden.

2.20 $HEXPATH

{$HEXPATH 'pathname'}

Group: CompilerSwitch 66

Werden mit "conditional compile" aus einer Source unterschiedliche Firmware Versionen erstellt, ist es sinnvoll, die generierten Hex-Files auch in unterschiedliche Directories abzulegen.
Alle beteiligten Tools, die aus der IDE heraus aufgerufen werden (Editor, Compiler, Assembler, Programmer) beachten diesen Schalter. Das Argument 'pathname' muss in Hochkomma dargestellt sein. Falls der Pfad bzw. Directory nicht existiert, wird der Pfad und Directory neu angelegt. Die beiden Schalter HexPath und HexName können auch in Kombination verwendet werden.

Bemerkung:
Eine Kopie der Flash und EEprom Hexfiles wird auch weiterhin in der Projekt Directory selbst abgelegt. Der InCircuit Programmer
kann dann direkt mit dem neuen bzw. geänderten Projekt aus der IDE heraus gestartet werden.

Im stand alone Betrieb des Programmers müssen dann allerdings die neuen Directories auch als neue Projekte angelegt werden.

2.21 $I

{$I fname}

Group: CompilerSwitch 66

Liest eine Include Datei, wobei "Filename" auch ein Pfad enthalten kann. Hiermit lassen sich Sourcen (Konserven), die immer wiederverwendet werden, einbinden. Die Include Datei kann sowohl Assembler als auch Pascal-Source als auch beides enthalten.
Die üblichen Konventionen des Compilers gelten hierbei natürlich weiterhin.

2.22 $IDATA

{$IDATA}

Group: CompilerSwitch 66

$IDATA weist allen nachfolgenden Variablen Deklarationen (dazu zählt auch **StructConst**) den im Prozessor Steuerfile (xxx.dsc) unter IDATA ausgewiesenen Bereich zu.

2.23 $IDATA1

{$IDATA1} **(*4*)**

Group: CompilerSwitch 66

Teilt das interne SRAM in zwei Teile.

```
Define iData1     = $800;
```

2.24 $IF

{$IF cond}

Group: CompilerSwitch 66

Hiermit können einfache bool'sche Ausdrücke angegeben werden. Als Operatoren sind nur "AND" und "OR" zulässig und
als Argumente nur solche die mit {$DEFINE ..} deklariert wurden.

{$IF ABC AND XYZ}
...
{$ELSIF HIJK OR OPQ}
...
{$ENDIF}

2.25 $IFDEF

{$IFDEF label}

Group: CompilerSwitch 66

Wenn "label" true ist, wird die nachfolgende Source compiliert bis zum "ELSE" bzw. "ENDIF". Ist "label" false, wird umgekehrt verfahren.

{$IFDEF CPUname}
Mit diesem Compiler Schalter kann CPU Typ abhängiger Code erzeugt werden.

{$IFDEF Mega128}
...
{$ENDIF}

2.26 $IFNDEF

{$IFNDEF name}

Group: CompilerSwitch 66

Wenn "label" false ist, wird die nachfolgende Source compiliert bis zum "ELSE" bzw. "ENDIF". Ist "label" true, wird umgekehrt verfahren.

2.27 $J

{$J fname}

Group: CompilerSwitch 66

Liest eine Include Datei, wobei "Filename" keinen Pfad enthalten darf. Als Pfad wird grundsätzlich die "Home-Directory" des Compilers vorangestellt.
Das ist sehr vorteilhaft für immer wiederkehrende Prozeduren etc. Und ersetzt fast den Linker bzw. das Unit-Konzept.

2.28 $LCDNOINIT

{$LCDNOINIT}

Group: CompilerSwitch 66

Wird das LCD Display wird beim Reset nicht initialisiert, muss das Anwendungs Programm dies selbst durch den Aufruf der System Prozedur "*LCDsetup*" machen.

2.29 $LCDNOWAIT

{$LCDNOWAIT}

Group: CompilerSwitch 66

Schaltet das Busy-Polling des Display Treibers ab. Nur für Debug Zwecke!

2.30 $MODBUS

{$MODBUS fname}

Group: CompilerSwitch 66

Bestimmt dass ein ModBus Debug File generiert wird, das mit dem ModBus Tester Tool verarbeitet werden kann.

2.31 $NOADDRCHECK

{$NOADDRCHECK}

Group: CompilerSwitch 66

Für Mega128..256. Schaltet die Bereichs Prüfung für die nachfolgend definierte Konstante ab. Diese Konstante kann ein externes binäres File sein das hier abgelegt wird. Liegt die Vorgabe Adresse ausserhalb der höchsten Flashpage (standard constant page), so ist ein Zugriff auf diese Konstante durch die Applikation nur mit ganz speziellen selbst zu erstellenden Verfahren möglich.

```
const
{$NOADDRCHECK}
LookUpTab [$20000] : array [1..256] of byte = 'Name.ext';          //mega2561
```

2.32 $NOFRAME

{$NOFRAME}

Group: CompilerSwitch 66

Gilt nur in Zusammenhang mit Device Treiber Prozeduren, die durch die Prozedur **Write** aufgerufen werden. Bei der nachfolgenden
Device Prozedur, die nur einen 8bit Übergabe Parameter besitzen darf, wird dieser Parameter in einem Register übergeben.
Ein Parameterframe wird nicht gebildet. Damit sind auch keine lokale Variablen möglich.
Nur für schnelle in Assembler geschriebene Treiber Routinen sinnvoll.

2.33 $NOINIT

{$NOINIT}

Group: CompilerSwitch 66

Schalter für den $DATA und den $IDATA Bereich. Die nachfolgenden Variablen dieses Bereiches bis zu dessen Ende
werden nicht initialisiert, d.h. nicht auf 0 gesetzt.

Man sollte jedoch dabei beachten, dass die Standard Initialisierung in einer sehr schnellen Schleife in einem Block erfolgt.
Werden jetzt noInit Variable beliebig dazwischen gestreut, kann dieses Init u.U. wesentlich langsamer ablaufen und auch mehr Code verbrauchen.
Man sollte also ggf. diese Var Definitionen zusammen fassen, so dass keine allzu grosse

Stückelung erfolgt.

2.34 $NOOVRCHECK

{$NOOVRCHECK}

Group: CompilerSwitch 66

Schaltet die Prüfung für die nachfolgende Variablen Overlay Deklaration ab.

2.35 $NORAMCHECK

{$NORAMCHECK}

Group: CompilerSwitch 66

Die nachfolgende Variable wird nicht auf einen gültigen Speicherbereich geprüft.

```
{$IDATA}
{$NORAMCHECK}
var
  Extreme[@$FFFF] : byte;
```

2.36 $NOREGSAVE

{$NOREGSAVE}

Group: CompilerSwitch 66

Gilt nur in Zusammenhang von Interrupt Prozeduren. Bei der nachfolgenden Interrupt Prozedur sichert das System absolut keine Arbeitsregister.
Der Programmierer muss selbst dafür sorgen. Nur für schnelle, in Assembler geschriebene Service Routinen sinnvoll.

2.37 $NORETURNCHECK

{$NORETURNCHECK}

Group: CompilerSwitch 66

Gilt nur in Zusammenhang von Funktionen. Bei der nachfolgenden Funktion erfolgt keine Fehlermeldung, wenn das Return Statement fehlt.

2.38 $NOSAVE

{$NOSAVE}

Group: CompilerSwitch 66

Gilt nur in Zusammenhang von Interrupt Prozeduren. Bei der nachfolgenden Interrupt Prozedur sichert das System die Arbeitsregister nicht automatisch, ausgenommen des Status Registers und der 4 Haupt Arbeitsregister.
Der Programmierer muss selbst dafür sorgen. Nur für schnelle, in Assembler geschriebene Service Routinen sinnvoll.

2.39 $NOSHADOW

{$NOSHADOW}

Group: CompilerSwitch 66

Die Definition muss, falls benötigt, noch vor der Device Deklaration erfolgen. Bei non-Multitask Anwendungen werden bei allen Interrupts
nur die durch die Interrupts benutzten Register gesichert. Dies spart wesentlich Ram, Rom und Rechenzeit.
Dieser Schalter wird durch den Import von Prozesse und Tasks überschrieben.

2.40 $NOWATCHDOGAUTO

{$NOWATCHDOGAUTO}

Group: CompilerSwitch 66

Compiler Schalter, wenn aktiv, dann erfolgen keine automatischen Watchdog Triggers in den Delays (mDelay etc),
als auch in den systembedingten Warteschleifen. Nur für besondere Applikationen.

2.41 $OPTIMISE

{$OPTIMISE}

Group: CompilerSwitch 66

Erzwingt das Einschalten des Optimiser.
Dieser Schalter **muss** in der ersten Zeile des Hauptprogramms stehen.

2.42 $OPTI ALLOW_INLINE

{$OPTI ALLOW_INLINE}

Group: CompilerSwitch 66

Schaltet die Ersetzung durch Inline Code für einfache Prozeduren (wie einfache "Setter" und "Getter") an. Nur für für Debug Zwecke.

siehe auch $OPTI NO_ALLOW_INLINE

2.43 $OPTI_BETA_OFF

{$OPTI_BETA_OFF}

Group: CompilerSwitch 66

Schaltet die in letzter Zeit eingeführten Optimierungen aus. Sollten Sie nur benutzen wenn Sie (wegen eines Bugs) dazu gezwungen sind.
Teilen Sie dies bitte Merlin mittels privater Mail im E-LAB Forum mit.

2.44 $OPTI CHECK_RETURN_REGS

{$OPTI CHECK_RETURN_REGS}

Group: CompilerSwitch 66

Bestimmt bei eigenem Assembler Code ob durch die .RETURNS Anweisung festlegt wurde, welche Register für die aufrufenden Funktion gerettet wurden oder ob weitere Register gesichert wurden.
Muss nach der jeweiligen 'end' Anweisung platziert werden

siehe auch $OPTI NO_CHECK_RETURN_REGS

2.45 $OPTI NO_ALLOW_INLINE

{$OPTI NO_ALLOW_INLINE}

Group: CompilerSwitch 66

Schaltet die Ersetzung durch Inline Code für einfache Prozeduren (wie einfache "Setter" und "Getter") aus. Nur für für Debug Zwecke.

siehe auch $OPTI ALLOW_INLINE

2.46 $OPTI NO_CHECK_RETURN_REGS

{$OPTI NO_CHECK_RETURN_REGS}

Group: CompilerSwitch 66

Bestimmt bei eigenem Assembler Code ob die .RETURNS Anweisung bestimmt, welche Register für die aufrufenden Funktion gerettet werden müssen oder ob weitere Register gesichert werden müssen.
Muss vor der jeweiligen Funktion/Prozedur stehen.

siehe auch $OPTI CHECK_RETURN_REGS

2.47 $OPTI NO_CSE_OPT

{$OPTI NO_CSE_OPT}

Group: CompilerSwitch 66

Schaltet die Optimierung für gemeinsame Ausgangspunkte ab.
Dies ist ein globaler Schalter der an beliebiger Stelle im Code stehen kann.
Bitte höchstens für Debug Zwecke benutzen!

2.48 $OPTI_QUICK

{$OPTI_QUICK}

Group: CompilerSwitch 66

Führt eine schnelle Optimierung durch. Weniger kompaktes Resultat,
aber eine leicht kürzere Compile Zeit.

2.49 $OPTI SMARTLINK_ONLY

{$OPTI SMARTLINK_ONLY}

Group: CompilerSwitch 66

Entfernt entfernt nur unbenutzte Funktionen/Prozeduren (dead code).
Alle anderen Optimiser Funktionen sind dann abgeschaltet.
Dieser Schalter arbeitet global, also auch für Units und Includes.

2.50 $OVERLAY

{$OVERLAY arg [, arg2]}

{$OverLay @VarName[, NoOvrCheck]}
{$OverLay 0}

Group: CompilerSwitch 66

Um das übereinander Mappen von Variablen nicht für jede Variable einzeln mit: *yyy[@xxx] : byte;* durchführen zu müssen.
Damit können z.B. mehrere Variablen in ein vorhandenes Array reingelegt werden.
VarName bezeichnet eine beliebige, existierende Variable (@VarName) im RAM Bereich. Der optionale Parameter "*NoOvrCheck*" bestimmt, dass keine Bereichsprüfung stattfinden soll.

Wird der Switch mit dem Parameter "0" übergeben, dann schliesst dieser den Overlay Bereich ab.
Alle Variablen, die zwischen den beiden Switches
platziert werden, erhalten jetzt aufsteigende Adressen ab "VarName". D.h. Sie werden auf die ursprüngliche Variable platziert, die normalerweise
in der Lage sein sollte alle neuen auch aufzunehmen. Ein Overflow wird beim abschliessenden Compiler Switch angezeigt, bzw. ignoriert
wenn die Option "*NoOvrCheck*" gesetzt wurde.

Statt eine Variablen als Basis kann auch eine absolute Adresse durch
{$OverLay $nnnn, NoOvrCheck}
angegeben werden. Die Angabe von "*NoOvrCheck*" ist hier dann zwingend.

2.51 $PCU

{$PCU}

Group: CompilerSwitch 66

Der in der IDE angesiedelte global innerhalb des Projekts wirkende Schalter "Project/Project Options" wirkt sich auf alle Units des Projekts aus.
Wenn aktiviert, werden alle Units des Projekts vorcompiliert und PCU Dateien werden erzeugt, abhängig von den weiteren Vorgaben ("copy" Schalter).

Durch Einsatz dieses Compiler Schalters im Source Bereich einer Unit wird die Generierung einer PCU für diese Unit erzwungen,
unabhängig davon wie der globale Schalter in der IDE steht. Die "copy" Schalter in der IDE sind auch hier wirksam.

2.52 $PDATA

{$PDATA}

Group: CompilerSwitch 66

$PDATA weist allen nachfolgenden Variablen Deklarationen den im Prozessor Steuerfile (xxx.dsc) unter PDATA ausgewiesenen Bereich zu.
Beim AVR 8515 ist dieser Bereich von $20 bis $5F. Die definierten Variablen werden jetzt fortschreitend ab $20 platziert.
PDATA ist für einen IO-Bereich reserviert, falls vorhanden. Variablen im Bereich $PDATA werden meistens mit speziellen Maschinenbefehlen erreicht.
Bei der Definition von Variablen in diesem Bereich lässt man nicht den Compiler die Adressen vergeben, sondern der Programmierer muss/sollte
zu jeder Variablen die gewünschte Adresse angeben:

***Var** Port1[$35] : byte;*

2.53 $PHASE

{$PHASE nnnn}

Group: CompilerSwitch 66

Schaltet auf feste **WORD**-Adresse im Flash um / zurück auf die Standard Code Page

{$PHASE $1E00}; legt den nachfolgenden Code ab Adr $1E00 ab.
{$DEPHASE} ; schaltet wieder um auf die Standard Code page.

2.54 $Q

{$Q-}

Group: CompilerSwitch 66

schaltet das Qualifizieren innerhalb des Assembler Codes ab.
Nur für schon vorhandene ältere Programme.

2.55 $REUTILIZE

{$REUTILIZE}

XMega

dient dazu dass Timer, SPI und TWI für verschiedene Treiber doppelt benutzt werden können.
Zulässige Argumente sind TIMER_C0..TIMER_F1, SPI_C..SPI_F, TWI_C..TWI_F

Group: CompilerSwitch 66

2.56 $SHOWERROR

{$SHOWERROR string}

Group: CompilerSwitch 66

erzeugt eine Fehlermeldung
{$ShowError 'Fehler bei ...'} erzeugt die Fehlermeldung beim Assemblieren

2.57 $SHOWWARNING

{$SHOWWARNING string}

Group: CompilerSwitch 66

erzeugt eine Warnung
{$ShowWarning 'Achtung: ...'} erzeugt die Warnung beim Assemblieren

2.58 $SL

{$SL+} / {$SL-}

zur Zeit abgeschaltet ! Bitte stattdessen den Merlin Optimiser benutzen !
siehe $OPTIMISE und $OPTI SMARTLINK_ONLY

Group: CompilerSwitch 66

Schaltet den Smart-Linker ein oder aus. Damit wird das Code löschen freigegeben (+) oder gesperrt (-). Diese Schalter können beliebig gesetzt werden.
Zu beachten ist dass der Schalter am Anfang einer Unit immer auf "off" gestellt ist. Das gleiche gilt für das Hauptprogramm File.

{$SL ON}
Schaltet beim Programmstart das Default Verhalten des Linkers auf aktiv.
Bei aktivem Schalter (on) kann eine Code Generierung für einzelne Funktionen oder Prozeduren auch erzwungen werden,
indem der Schalter *{$VALIDATE ProzedurName}* eingesetzt wird.

Dieser Schalter wird aber erst wirksam, nachdem diese Funktion schon dem System bekannt, d.h. definiert ist.
Eine Alternative dazu ist vor einer Funktion/Prozedur Deklaration der Schalter *{$VALIDATE $}*.

2.59 $TYPEDCONST

{$TYPEDCONST ON}
{$TYPEDCONST OFF}

Group: CompilerSwitch 66

Für eine bessere Lesbarkeit des Programms und zur Vermeidung von Compiler Fehlern. Mit "ON" erwartet der Compiler mit jeder
Konstanten Deklaration auch die zugehörige Typ Deklaration. Damit wird z.B. eine "0" auch eindeutig entweder ein Byte, Word, Integer oder Float.

```
const bb : byte = 0;
```

Der Schalter ist default "ON".
Wenn man vorhandene Programme nicht umschreiben will, so sollte gleich beim Programm Beginn der Schalter so verwendet werden:

```
{$TYPEDCONST OFF}
program ProgName;
```

Die Option "OFF" sollte nicht mehr verwendet werden!

2.60 $UDATA

{$UDATA}

Group: CompilerSwitch 66

$UDATA weist allen nachfolgenden Variablen Deklarationen den im Definitionsteil deklarierten UserData (UserDevice) Bereich zu.
Dieser Datenbereich liegt in einem externen Device, das nicht über die normale CPU-Adressierung angesprochen werden kann,
z.B. serielles Eeprom. Der Programmierer muss dazu einen Device Treiber bereitstellen.
Siehe auch Abschnitt **Device Treiber** im *Standard Driver Manual*.

Die Variablen bauen sich immer von den niederen Adressen zu den höheren auf. Wechseln die Speicherbereiche durch einen neuen Schalter,
so wird mit dem aktuellen Speicher an der **zuletzt vergebenen Adresse** fortgefahren.

2.61 $UNDEF

{$UNDEF label}

Group: CompilerSwitch 66

Setzt "label" auf false

2.62 $VALIDATE

{$VALIDATE name}

Group: CompilerSwitch 66

Konstante und Variable des Systems als auch der Applikation werden vom Compiler in der Regel "wegoptimiert", wenn diese nicht im Kontext angesprochen werden, d.h. wenn sie nicht irgendwo in einem Statement auftauchen. Das kann zu Problemen beim Inline Assembler Code in der Pascal Source führen. Der Assembler gibt dann einen Fehler aus, da die angesprochene Prozedur etc. nicht vorhanden ist. Mit diesem Schalter wird der Compiler gezwungen, auf jeden Fall das Konstrukt "name" zu importieren bzw. die Optimierung greift hier nicht.
ACHTUNG:
diese Option funktioniert (als einzige) nicht mit dem *"Merlin Optimiser"*

{$VALIDATE $}

Group: CompilerSwitch 66

Funktionen und Prozeduren können durch den vorangestellten Schalter von jedweder Optimierung ausgeschlossen werden.

2.63 $VALIDATE_ALL

{$VALIDATE_ALL}

Group: CompilerSwitch 66

Damit wird der Compiler veranlasst auch offensichtlich im Programm nicht benutzte Konstante vom Typ String, Array und Record
im Programmcode abzulegen. Die Optimierung für diese Art der Konstanten ist damit komplett abgeschaltet.

2.64 $VALIDATE_OFF

{$VALIDATE_OFF}

Group: CompilerSwitch 66

damit werden ganze Blöcke von Variablen als "benutzt" gekennzeichnet, so dass die Meldung "possibly unused variable" für diesen Bereich nicht auftaucht.

{$VALIDATE_ON}

```
VAR
  ...
  ...
{$VALIDATE_OFF}
```

2.65 $VALIDATE_ON

{$VALIDATE_ON}

Group: CompilerSwitch 66

damit werden ganze Blöcke von Variablen als "benutzt" gekennzeichnet, so dass die Meldung "possibly unused variable" für diesen Bereich nicht auftaucht.

```
{$VALIDATE_ON}
VAR
  ...
  ...
{$VALIDATE_OFF}
```

2.66 $VectTab

*{$VectTab $nnnn}* // word address, z.B. {$VectTab $0F000}

Group: CompilerSwitch 66

Legt den Anfang der Interrupt Vector Tabelle fest. Nur für ganz spezielle Aufgaben, wie das Erstellen einer Applikation, die direkt im BootBlock platziert wird.
siehe auch: $BootApplication und $CodeStart

2.67 $W

{$W+} / {$W-}

Group: CompilerSwitch 66

Schaltet die Überwachung von Variablen und Prozeduren in Main oder Units ein. Bei offensichtlich unbenutzten Variablen oder
Prozeduren erzeugt der Compiler dabei eine Warnung, die durch die IDE ausgewertet wird. (siehe auch Optimierung und $WG)

Default: {$W-}

2.68 $WG

{$WG}

Group: CompilerSwitch 66

Schaltet die Überwachung von Variablen und Prozeduren für alle Programmteile/Units ein. Bei

offensichtlich unbenutzten Variablen
oder Prozeduren erzeugt der Compiler dabei eine Warnung, die durch die IDE ausgewertet wird.
(siehe auch Optimierung und $W)

Default: ausgeschaltet

2.69 $X

{$X+} / {$X-}

Group: CompilerSwitch 66

Die nachfolgenden Statements werden durch den Simulator nicht ausgeführt. Hilfreich bei sehr langen mDelays während des Debuglaufs
oder wenn auf externe Hardware gewartet wird.

Dieser Schalter hat keinerlei Einfluss auf das generierte Hexfile, d.h. er muss nicht entfernt werden.

2.70 $XDATA

{$XDATA} / {$XDATA1} / {$XDATA2} / {$XDATA3} / {$XDATA4}

Group: CompilerSwitch 66

$XDATA weist allen nachfolgenden Variablen Deklarationen (dazu zählt auch **StructCons**) den mit **Define** XDATA ausgewiesenen Bereich zu.
XDATA ist externer Speicher, der nur bei den grösseren Typen anzutreffen ist. Folgt ein anderer Schalter dieses Typs, wird mit den folgenden Variablen analog dazu verfahren. Variablen im Bereich $XDATA werden immer mit längeren und damit langsameren Maschinenbefehlen erreicht. Oft setzt die CPU noch zusätzliche Waitstates ein.

2.71 $XIO

{$XIO+} / {$XIO-}

Group: CompilerSwitch 66

Kann im XDATA Bereich verwendet werden um dem Optimierer z.B. Memory Mapped IO-Bereiche zu signalisieren.
Der Bereich innerhalb dieser beiden Schalter wird getrennt behandelt bzw. redundante Zugriffe werden nicht entfernt.

2.72 $ZEROLOCVARS

{$ZEROLOCVARS}

Group: CompilerSwitch 66

Wenn aktiv, werden alle lokalen Variablen in Funktionen und Prozeduren auf 0 gesetzt wenn eine Funktion oder Prozedur aufgerufen wird.

2.73 Einleitung

Liste: Compiler Switch 3

Compilerschalter dienen dazu das Verhalten des Compilers zu steuern. Diese Schalter sind Bestandteil des Quelltextes/Source. Ein Schalter wird durch eine geschweifte Klammer **{** und einem darauf ohne Leerzeichen folgenden **$** eingeleitet. Unmittelbar auf den $ muss der Schaltername folgen und zwar in Grossbuchstaben. Die Angabe evtl. weiterer Parameter wie z.B. Filenamen erfolgen nach den üblichen Konventionen.

Die Schalter müssen unbedingt in einer Zeile am Zeilen Anfang stehen. In der gleichen Zeile dürfen keine Statements stehen.
Syntax: **{$SWITCH [arg] }**
Jeder Import und jedes Device Define wird auch in die Compiler Schalter Liste eingefügt. Das bedeutet dass:

Import *LCDport, ...*

behandelt wird als wäre es: {$DEFINE LCDPORT} und

Define *ProcClock = 8000000;*

wird behandelt wie: {$DEFINE PROCCLOCK}

Conditional Compile
Es ist manchmal notwendig, aus einem Programm unterschiedliche, z.B. hardware-abhängige, Versionen zu generieren. Das jeweilige Verhalten des Compilers kann dazu mit Hilfe der Compiler-Schalter für abhängige Compilation (Conditional Compile) gesteuert werden. Das dazu verwendete "***Label***" hat dabei nur symbolischen Charakter. Ist das Ergebnis eines Schalters "falsch", so wird der ab hier beginnende Source-Code bis zum "wahr"-werden des Schalters als Kommentar behandelt bzw. existiert nicht. Alle Schalter dieser Gruppe können an beliebiger Stelle in der Source vorkommen.

Eine "IFxx"-Anweisung muss immer mit einem "ENDIF" abgeschlossen sein. Dazwischen kann ein "ELSE" liegen. Verschachtelte Compilerschalter sind ebenfalls zulässig!!
LABELS können auch in der IDE PED32 unter ***Project/Project Options*** definiert werden. Mehrere Labels müssen durch Strichpunkte getrennt werden. Hierbei dürfen jedoch nur die "nackten" Labels stehen ohne $ und ohne DEFINE etc. PED32 informiert den Compiler über die DEFINES und dieser behandelt sie so, als wären sie direkt in der Source in der ersten Zeile.

3 Funktionen + Prozeduren

3.1 Abs

Function *Abs (i : integer) : integer;*
Function *Abs (f : float|fix64) : float|fix64;*
Absolutwert eines Int8, Integer, Longint, Int64, Fix64 oder Float Wertes

Group: Maths 5

3.2 AddAVFilter

Procedure *AddAVfilter (**var** Filter : AVfilter; val : type);*
Ersetzt den ältesten Wert ohne neue Mittelwert Bildung.

Group: Diverse 4

3.3 Addr

Function *Addr (identifier) : pointer; {Adresse der Speicherstelle}*
Als Operanten sind nur Variablen, Prozeduren und Funktionen zulässig.
Das Resultat der Funktion ist ein **typisierter** Pointer.

Group: System 7

3.4 Append

Procedure *Append (src : string; **var** dst : string);*
Hängt den String "src" an den String "dst" an (concat).

Group: Strings 6

3.5 ArcTan

Function *ArcTan (w : float) : float;*
Liefert als Ergebnis den Arkustangens des Arguments.

Group: Maths 5

3.6 ArrToStr

Function *ArrToStr (arr : **array of** char) : string;*
Konvertiert einen Null-terminierten String zu einem Pascal String.

Group: Strings 6

3.7 AVR_CAN_BaudRate

Function *AVR_CAN_BaudRate (br : tCAN_baud) : boolean;*
Stellt die Baudrate ein. Zuvor sollte ein Disable und danach ein Enable ausgeführt werden.

Group: AVR-CAN 8

3.8 AVR_CAN_Disable

Procedure *AVR_CAN_Disable;*
Sperrt den Treiber so dass die Applikation Änderungen vornehmen kann.

Group: AVR-CAN 8

3.9 AVR_CAN_Enable

Procedure *AVR_CAN_Enable;*
Gibt den Treiber nach einem Disable wieder frei.

Group: AVR-CAN

3.10 AVR_CAN_GetError

Function *AVR_Can_GetError (box : byte) : boolean;*
Prüft den aktuellen Status der Box auf aufgetretene Fehler.

Group: AVR-CAN

3.11 AVR_CAN_GetStatus

Function *AVR_Can_GetStatus (box : byte) : tAVR_CAN_States;*
Gibt den aktuellen Status der Box zurück.

Group: AVR-CAN

3.12 AVR_CAN_Init

Function *AVR_CAN_Init (RxMObCount : Byte) : boolean;*
Setzt die CAN Hardware komplett zurück und initialisiert diese neu.
RxMObCount (1..14) bestimmt die Anzahl der für den Empfang zu initialisierenden Mailboxen.

Achtung: diese Funktion wurde mit Compiler Rev. 4.92.00 geändert und muss ggf. angepasst werden.

Group: AVR-CAN

3.13 AVR_CAN_RxErrCount

Function *AVR_CAN_RxErrCount : byte;*
Gibt Rx Error Count zurück.

Group: AVR-CAN

3.14 AVR_CAN_SetRxEMask

Function *AVR_CAN_SetRxEMask (box : byte; lwIDtag, lwIDmask : longword) : boolean;*
Stellt den erweiterten ID-TAG und die ID-Mask für eine Rx-Mailbox (1..14) ein.

Group: AVR-CAN

3.15 AVR_CAN_SetRxMask

Function *AVR_CAN_SetRxMask (box : byte; wIDtag, wIDmask : word) : boolean;*
Stellt den Standard ID-TAG und die ID-Mask für eine Rx-Mailbox (1..14) ein.

Group: AVR-CAN 8

3.16 AVR_CAN_StartMessage

Procedure *AVR_CAN_StartMessage;*
Diese Funktion muss aufgerufen werden um das Senden freizugeben.

Group: AVR-CAN 8

3.17 AVR_CAN_TxErrCount

Function *AVR_CAN_TxErrCount : byte;*
Gibt Tx Error Count zurück.

Group: AVR-CAN 8

3.18 BankDevPtr

Function *BankDevPtr (b:byte; p:pointer): pointer;*
Pointer zeigt als Ergebnis ins Banked Device.

Group: Banking Port 9

3.19 BCDtoByte

Function *BCDtoByte (b : byte) : byte;*
Konvertiert einen packed BCD Wert in ein Byte.

Group: Maths 5

3.20 BeepChirpH

Procedure *BeepChirpH (repTimes : byte);*
Erzeugt eine sehr kurze absteigende hohe Tonfolge (Chirp).

Group: BeepPort 9

3.21 BeepChirpL

Procedure *BeepChirpL (repTimes : byte);*
Erzeugt eine sehr kurze absteigende niedrige Tonfolge (Chirp).

Group: BeepPort 9

3.22 BeepClick

Procedure *BeepClick;*
Erzeugt ein kurzes Klicken.

Group: BeepPort 9

3.23 BeepOut

Procedure *BeepOut (Frequ : word; ticks : byte);*
Diese Prozedur erzeugt einen Ton mit der angegeben Frequenz.

Group: BeepPort 9

3.24 BeepOutErr

Procedure *BeepOutErr;*
Erzeugt einen kurzen schnarrenden Ton.

Group: BeepPort 9

3.25 BeepOutHL

Procedure *BeepOutHL;*
Erzeugt eine kurze absteigende Tonfolge mit 3 Tonfolgen.

Group: BeepPort 9

3.26 BeepOutLH

Procedure *BeepOutLH;*
Erzeugt eine kurze aufsteigende Tonfolge mit 3 Tonfolgen.

Group: BeepPort 9

3.27 BeepSiren

Procedure *BeepSiren (**const** mode : byte; repTimes : byte);*
Erzeugt einen Sirenzyklus.

Group: BeepPort 9

3.28 BeepStepHL

Procedure *BeepStepHL;*
Erzeugt eine kurze absteigende Tonfolge mit 5 Tonfolgen.

Group: BeepPort 9

3.29 BeepStepLH

Procedure *BeepStepLH;*
Erzeugt eine kurze aufsteigende Tonfolge mit 5 Tonfolgen.

Group: BeepPort 9

3.30 Bit

1. Vordefinierter Type. Benötigter Speicher: 1bit, true..false, 0..1

```
const LedBit2 : byte = 3;
var   port6 : byte;
      Led2[@port6, Led2Bit] : bit;
```

Group: Reserved Words 6

2. Testet ein bit auf true/false.
Function *Bit (a, 0 : byte : boolean;*
Function *Bit (b : bit) : boolean;*

Group: System 7

3.31 BitCountOf

Function *BitCountOf (x : ordinal) : byte;*
Gibt die Anzahl der gesetzten Bits zurück.

Group: Diverse 4

3.32 BoolToStr

Function *BoolToStr (bool : boolean) : string;*
Function *BoolToStr (bool : boolean; TrueStr, FalseStr : string) : string;*
Konvertiert eine Boolean Variable in einen String.

Group: Strings 6

3.33 Boot_Init

Procedure *Boot_Init;*
Stellt den Stack und Frame so bereit wie es im Define vorgegeben wurde.

Group: System 7

3.34 BootRestart

Procedure *BootRestart;*
Die Applikation kann hiermit einen Bootvorgang auslösen.

Group: System 7

3.35 ByteToBCD

Function *ByteToBCD (b : byte) : byte;*
Konvertiert ein Byte in das BCD Format.

Group: Maths 5

3.36 ByteToBin

Function *ByteToBin (value : byte|int8) : string;*
Das Ergebnis ist die Repräsentation der einzelnen Bits durch eine '0' oder '1' im String.

Group: Strings 6

3.37 ByteToHex

Function *ByteToHex (b : byte|int8) : string;*
Konvertiert Numerischen 8bit Wert in einen hex-String.

Group: Strings 6

3.38 ByteToStr

Function *ByteToStr (b : byte|int8) : string;*
Konvertiert Numerischen 8bit Wert in einen String.

Group: Strings 6

3.39 CalcCheckSum

Function *calcCheckSum (**const** start, end : pointer) : word;*
Bildet die Summe aller Bytes im angegebenen EEProm Bereich.

Group: Maths 5

3.40 CalcFlashCheck

Function *CalcFlashCheck : boolean;*
bildet zur Laufzeit eine Checksumme und vergleicht diese mit der im Flash abgelegten.

Group: System 7

3.41 CalcFlashCheck_A

Function *CalcFlashCheck_A (count : word) : byte;*
überprüft die Applikations Checksumme aus dem Bootbereich heraus.

Der Übergabe Parameter der Funktion bestimmt die Anzahl der Bytes,
die in diesem Durchgang geprüft werden sollen.
Die Funktion gibt eine 0 zurück wenn das Check Ende in diesem Durchgang noch
nicht erreicht wurde. Ist das Ende erreicht, so wird das Ergebnis ungleich Null:
1 = check finished and ok

2 = check finished but failed.

muss importiert werden:
***From** System **Import** FlashCheck_A;*

Group: System

3.42 CalcFlashCheck_B

***Function** CalcFlashCheck_B : boolean;*
Bildet einen separaten Flash Check über den ganzen Bootbereich.

muss importiert werden:
***From** System **Import** FlashCheck_B;*

Group: System

3.43 CalcFlashCheck_S

***Function** CalcFlashCheck_S (count : word) : byte;*

muss importiert werden:
***From** System **Import** FlashCheck_S;*

sequentieller (partieller) Flash Check. Aufruf bis Ergebnis <> 0.

Group: System

3.44 ChangeDir

***Function** ChangeDir (**const** dir : char) : boolean; // '0'..'9'*
Stellt die *DefaultDirectory* für alle Drives ein.

Group: 8bit FileSystem

3.45 CheckFrameValid

***Function** CheckFrameValid (p : Process|Task) : integer;*
Ermittelt ob ein Frame Überlauf stattgefunden hat.

Group: System

3.46 CheckStackValid

***Function** CheckStackValid (p : Process|Task) : integer;*
Ermittelt ob ein Stack Überlauf stattgefunden hat.

Group: System

3.47 ClearDeviceLock

Function *ClearDeviceLock (d : DeviceLock) : boolean;*
Gibt den Treiber wieder frei.

Group: MultiTasking 5

3.48 ClearIncrAll4

Procedure *ClearIncrAll4;*
Setzt alle absoluten und relativen internen Zähler auf null.

Group: Increment Counter 4chan 15

3.49 ClearIncrementVal

Procedure *ClearIncrementVal;*
Setzt den absoluten und relativen internen Zähler auf null.

Group: Increment Counter 15

3.50 ClearIncrVal4

Procedure *ClearIncrVal4 (chan : byte);*
Setzt den absoluten und relativen internen Zähler von *chan* auf null.

Group: Increment Counter 4chan 15

3.51 ClearKeyBoard

Procedure *ClearKeyBoard;*
Setzt das komplette KeyBoard zurück.

Group: KeyBoard 4x4 16

3.52 ClearKeyBoard8

Procedure *ClearKeyBoard8;*
Setzt das komplette KeyBoard zurück.

Group: KeyBoard 8x8 17

3.53 ClearRunErr

Procedure *ClearRunErr;*
Setzt einen aufgetretenen Laufzeitfehler zurück.

Group: System 7

3.54 CompareBlock

Function *CompareBlock (**const** addr1, addr2 : pointer; **const** len : word) : boolean;*
Vergleicht zwei Speicherbereiche miteinander.

Group: System 7

3.55 CompareIP

Function *CompareIP (ip1, ip2 : TIPAddr) : boolean;*
Vergleicht zwei IP-Adress Arrays.

Group: System 7 TINA TCP/IP 41 WizNet TCP/IP 45

3.56 CompareMAC

Function *CompareMAC (mac1, mac2 : TMACAddr) : boolean;*
Vergleicht zwei MAC-Adress Arrays.

Group: System 7 TINA TCP/IP 41 WizNet TCP/IP 45

3.57 CompareNet

Function *CompareNet (a1, a2, mask : TIPAddr) : boolean;*
Vergleicht den Netzwerk-Teil zweier IP-Adress Arrays.

Group: System 7 TINA TCP/IP 41 WizNet TCP/IP 45

3.58 Copy

Function *Copy (st : string; pos, count : byte) : string;*
Kopiert aus einem String einen Teilstring. Das Ergebnis ist ein String.

Group: Strings 6

3.59 CopyBlock

Procedure *CopyBlock (src : pointer; dst : pointer; len : word);*
Kopiert einen Speicherbereich in einen anderen.

Group: System 7

3.60 Cos

Function *Cos (w : float) : float;*
Liefern den Cosinus des Arguments zurück.

Group: Maths 5

3.61 CosD

Function *CosD (w : float) : float;*
Liefern den Cosinus des Winkels w im Gradmaß zurück.

Group: Maths 5

3.62 CosInt

Function *CosInt (angle, v : integer) : integer;*
Die Funktionen liefern den Cosinus des Winkels multipliziert mit v.

Group: Maths 5

3.63 CosInt16

Function *CosInt16 (angle : integer) : integer; // angle in 0.1deg*
Errechnet den Cosinus des Winkels, multipliziert mit 10000.

Group: Maths 5

3.64 CPUsleep

Procedure *CpuSleep (sleepcmd : byte);*
Legt die CPU schlafen, d.h. das Programm wird gestoppt.

Group: System 7

3.65 CRCcheck

Function *CRCcheck (p : pointer; count : word) : word;*
Prüft Datenblock mit dem angehängten CRC-Word.

Group: System 7

3.66 CRCstreamAdd

Function *CRCstreamAdd (value : byte) : word;*
Fügt kontinuierlich Bytes (value) der Checksumme hinzu.

Group: System 7

3.67 CRCstreamAddP

Function *CRCstreamAddP (ptr : pointer; count: word) : word;*
Fügt der Checksumme ganze Speicherblöcke (RAM) hinzu.

Group: System 7

3.68 CRCstreamInit

Procedure *CRCstreamInit (seed : word);*
Setzt den CRC Start Wert auf 0.

Group: System 7

3.69 DCFdayLightSave

Function *DCFdayLightSave : boolean;*
Gibt die Sommer/Winterzeit zurück. Sommerzeit = true.

Group: DCF-77 Encoder 10

3.70 DCFfield

Function *DCFfield : byte;*
Gibt einen ungefähren Wert für die Empfangsqualität wieder.

Group: DCF-77 Encoder 10

3.71 DCFready

Function *DCFready : boolean;*
Zeigt an ob einmal ein Telegramm empfangen wurde.

Group: DCF-77 Encoder 10

3.72 DCFupdate

Procedure *DCFupdate; // Callback Procedure*
Findet der DCF77 Treiber den Import der RTC nicht, so ruft er jede Minute die Prozedur „DCFupdate" auf.

Group: DCF-77 Encoder 10

3.73 DDS10buildTab

Procedure *DDS10buildTab (DDStab, amp : byte; dsMode : tdsMode);*
Initialisiert eine DDS10 Lookup Tabelle.

Group: DDS10 Synthesizer 10

3.74 DDS10setFrequ

Procedure *DDS10setFrequ (freq : float);*
Stellt die aktuelle Frequenz ein.

Group: DDS10 Synthesizer 10

3.75 DDS10setTab

Procedure *DDS10setTab (DDStab : byte);*
Schaltet zur Laufzeit zwischen den vorhandenen Tabellen um.

Group: DDS10 Synthesizer 10

3.76 DDS10start

Procedure *DDS10start;*
Startet eine evtl. abgeschaltete Synthese wieder.

Group: DDS10 Synthesizer 10

3.77 DDS10stop

Procedure *DDS10stop;*
Stoppt eine Synthese.

Group: DDS10 Synthesizer 10

3.78 Dec

Procedure *Dec (**var** v [, step] : type);*
Dekrementiert eine Variable.

Group: Maths 5

3.79 DecIAVfilter

Function *DecIAVfilter (**var** Filter : AVfilter) : type;*
Errechnet die Steigung zwischen dem ältesten und dem jüngsten Eintrag.

Group: Diverse 4

3.80 DecSema

Function *DecSema (s : semaphore) : boolean;*
Eine Semaphore wird um eins erniedrigt.

Group: System 7

3.81 DecToLim

Function *DecToLim (**var** v : ordinal [, limit : ordinal[; val : ordinal]]) : boolean;*
Das Argument "v" wird um 1 erniedrigt.

Group: Maths 5

3.82 DecToLimWrap

Function *DecToLimWrap (**var** value, lim, pres : type) : boolean;*
Dekrementiert die Variable "value" jeweils um 1.

Group: Maths 5

3.83 DegToRad

Function *DegToRad (w : float) : float;*
Wandelt eine im Gradmaß vorliegende Winkelgröße in Bogenmaß um.

Group: Maths 5

3.84 Delete

Procedure *Delete (**var** s : string; pos, count : byte);*
Löscht eine Anzahl Zeichen in einem String.

Group: Strings 6

3.85 Disable_JTAGport

Procedure *Disable_JTAGport;*
Die Applikation kann zur Laufzeit das JTAG Port ausschalten.

Group: System 7

3.86 DisableInts

Procedure *DisableInts;*
Sperrt den globalen Interrupt.

Group: System 7

3.87 DiskFormat

Function *DiskFormat (**const** drive : char) : boolean; // 'A'..'D'*
Der Directory Teil des Drives wird initialisiert bzw. komplett gelöscht.

Group: 8bit FileSystem 7

3.88 DiskFree

Function *DiskFree (**const** dir : char) : word; // 'A'..'D'*
Gibt den noch freien Speicherplatz auf dem Drive zurück (in kBytes).

Group: 8bit FileSystem 7

3.89 DiskReset

Function *DiskReset (**const** drive : char) : boolean; // 'A'..'D'*
Setzt das Laufwerk zurück.

Group: 8bit FileSystem 7

3.90 DiskSelect

Function *DiskSelect (**const** drive : char) : boolean; // 'A'..'D'*
Stellt das DefaultDrive ein.

Group: 8bit FileSystem 7

3.91 Disp14Blink

Procedure *Disp14Blink (**const** blink : boolean);*
Das ganze Display blinkt bzw. das Blinken wird ausgeschaltet.

Group: LED14seg Display 22

3.92 Disp14Clear

Procedure *Disp14Clear;*
Das ganze Display wird gelöscht.

Group: LED14seg Display 22

3.93 Disp14ClrEOL

Procedure *Disp14CLREOL;*
Das Display wird von der aktuellen Cursorposition bis Zeilenende gelöscht.

Group: LED14seg Display 22

3.94 Disp14DigBlink

Procedure *Disp14DigBlink (digit: byte; blink: boolean);*
Ein einzelnes Digit blinkt bzw. das Blinken wird ausgeschaltet.

Group: LED14seg Display 22

3.95 Disp14Out

Procedure *Disp14Out (**const** ch : char);*
Schreiben in den Display Buffer.

Group: LED14seg Display 22

3.96 Disp14Pos

Procedure *Disp14Pos (**const** digit : byte);*
Der Schreibcursor wird an eine bestimmte Stelle positioniert.

Group: LED14seg Display 22

3.97 Disp14Test

Procedure *Disp14Test;*
Alle Segmente des Displays werden eingeschaltet.

Group: LED14seg Display 22

3.98 Disp7Test

Procedure *Disp7Test;*
Testfunktion. Schaltet alle Segmente ein.

Group: LED7seg Display 23

3.99 DispBlink

Procedure *DispBlink (true);*
Das ganze Display blinkt bzw. das Blinken wird ausgeschaltet.

Group: LED7seg Display 23

3.100 DispClear

Procedure *DispClear;*
Das ganze Display wird gelöscht.

Group: LED7seg Display 23

3.101 DispClEOL

Procedure *DispClEOL;*
Das Display wird von der aktuellen Cursorposition bis Zeilenende gelöscht.

Group: LED7seg Display 23

3.102 DispDigBlink

Procedure *DispDigBlink (x : byte);*
Ein einzelnes Digit blinkt bzw. das Blinken wird ausgeschaltet.

Group: LED7seg Display 23

3.103 DispOut

Procedure *DispOut (ch : char);*
Schreiben in den Display Buffer.

Group: LED7seg Display 23

3.104 DispPos

Procedure *DispPos (x : byte);*
Der Schreibcursor wird an eine bestimmte Stelle positioniert.

Group: LED7seg Display 23

3.105 EdBoolean

Function *EdBoolean (EdValue : Boolean; LeadLabel, PostLabel : String[EdLabelLength];*
X, Y : Byte; BlinkCursor : Boolean) : Boolean;

Boolean editieren.

Beschreibung der Parameter bei Group: LCD Edit 19

3.106 EdByte

Function *EdByte (EdValue : Byte; LeadLabel, PostLabel : String[EdLabelLength];*
X, Y : Byte; BlinkCursor : Boolean; VMin, VMax : Byte) : Byte;

Byte editieren.

Beschreibung der Parameter bei Group: LCD Edit 19

3.107 EdDate

Function *EdDate (EdValue : String[EdDateLength]; LeadLabel : String[EdLabelLength];*
X, Y : Byte; BlinkCursor : Boolean) : String[EdDateLength];

Datum editieren.

Beschreibung der Parameter bei Group: LCD Edit 19

3.108 EdInteger

Function *EdInteger (EdValue : Integer; LeadLabel, PostLabel : String[EdLabelLength];*
X, Y : Byte; BlinkCursor : Boolean; VMin, VMax : Integer;
Decimal : Byte) : Integer;

Integer editieren.

Beschreibung der Parameter bei Group: LCD Edit 19

3.109 EdIPaddress

Function *EdIPAddress (EdValue : tEDIPAddress; LeadLabel, PostLabel : String[EdLabelLength];*
X, Y : Byte; BlinkCursor : Boolean;
IPMin, IPMax : tEdIpAddress) : tEDIPAddress;

IP Adressen editieren.

Beschreibung der Parameter bei Group: LCD Edit 19

3.110 EdList

Function *EdList (EdValue : Pointer; Location : tEdArrayLocation;*
LeadLabel, PostLabel : String[EdLabelLength]; X, Y : Byte;
BlinkCursor : Boolean; StrLen, Count, Default : Byte) : Byte;

Anzeige von String Listen, darin auf- und abwärts blättern und einen Eintrag auswählen.
Bitte beachten, dass das Array mit dem Index 0 beginnen muss.

EdValue	ist hier ein Pointer auf ein beliebiges Array of string (siehe Demoprogramm)
Location	bestimmt den auf den Speichertyp (RAM, ROM, EEPROM) des Array's
StrLen	ist die maximale Länge eines Strings im Array (Achtung auf die Länge achten! Hier wird mit Pointern gearbeitet)
Count	ist die Anzahl der Einträge im Array (Achtung auf die Anzahl achten! Hier wird mit Pointern gearbeitet.)

Group: LCD Edit 19

3.111 EdLongInt

Function *EdLongInt (EdValue : LongInt; LeadLabel, PostLabel :String[EdLabelLength];*
X, Y : Byte; BlinkCursor : Boolean; VMin, VMax : LongInt;
Decimal : Byte) : LongInt;

LongInt editieren.

Beschreibung der Parameter bei Group: LCD Edit 19

3.112 EdLongWord

Function *EdLongWord (EdValue : LongWord; LeadLabel, PostLabel : String[EdLabelLength];*
X, Y : Byte; BlinkCursor : Boolean; VMin, VMax : LongWord;
Decimal : Byte) : LongWord;

LongWord editieren.

Beschreibung der Parameter bei Group: LCD Edit 19

3.113 EdString

Function *EdString (EdValue : String[EdStringLength];*
LeadLabel, PostLabel String[EdLabelLength];
X, Y : Byte; BlinkCursor : Boolean; MaxLen : Byte;
MinChar, MaxChar : Byte;
Repeater : Boolean) : String[EdStringLength];

String editieren.
MaxLen bestimmt die maximale Länge des zu editierenden Strings

Beschreibung der Standard Parameter bei Group: LCD Edit 19

3.114 EdTime

Function *EdTime (EdValue : String[EdTimeLength]; LeadLabel : String[EdLabelLength];*
X, Y : Byte; BlinkCursor : Boolean;
EditSeconds : Boolean) : String[EdTimeLength];

Zeit editieren.
EditSeconds bestimmt, ob im Edit-Feld auch die Sekunden editiert werden können

Beschreibung der Standard Parameter bei Group: LCD Edit 19

3.115 EdWord

Function *EdWord (EdValue : Word; LeadLabel, PostLabel : String[EdLabelLength]; X, Y : Byte;*
BlinkCursor : Boolean; VMin, VMax : Word; Decimal : Byte) : Word;

Word editieren.

Beschreibung der Parameter bei Group: LCD Edit 19

3.116 EEpromPtr

Function *EEPromPtr (p:pointer): pointer;*
Lässt den Pointer ins EEprom zeigen.

Group: System 7

3.117 Enable_JTAGport

Procedure *Enable_JTAGport;*
Die Applikation kann zur Laufzeit das JTAG Port einschalten.

Group: System 7

3.118 EnableInts

Procedure *EnableInts;*
Gibt den globalen Interrupt frei.

Xmega
erweitert mit einem Parameter, der die enabled/disabled Interrupt Levels definiert..
Procedure *EnableInts (level : byte);*

Group: System 7

3.119 EnableIntsX

Procedure *EnableIntsX;*

Xmega
gibt den globalen Xmega Interrupt frei ohne die Levels zu ändern.

Group: System 7

3.120 EnablePWM

procedure *EnablePWM_C0A (ena : boolean);*
procedure *EnablePWM_C0B (ena : boolean);*
procedure *EnablePWM_C0C (ena : boolean);*
. . .

nur XMega
Gibt den jeweiligen PWM Port frei oder sperrt ihn.

Group: PWM Port 27

3.121 EndOfFile

Function *EndOfFile (**const** f : file) : boolean;*
Stellt fest ob der Lese/Schreib Pointer am Dateiende angelangt ist.

Group: 8bit FileSystem 7

3.122 Even

Function *Even (x : type) : boolean;*
Wert auf geradzahlig testen.

Group: Maths 5

3.123 Excl

Procedure *Excl (v, b : byte);*
Bit in einem Byte rücksetzen.

Procedure *Excl (SrcDest : BitSet; op : BitSet);*

Ein BitSet in einem BitSet rücksetzen.

Group: System 7

3.124 Exp

Function *Exp (x : float) : float;*
Gibt die Potenz von x zurück.

Group: Maths 5

3.125 ExtractFileExt

Function *ExtractFileExt (FName : string) : string;*
Liefert den Extension-Teil innerhalb eines FileNamens zurück.

Group: Strings 6

siehe auch FAT16_File_System 11

3.126 ExtractFileName

Function *ExtractFileName (FName : string) : string;*
Liefert den Namens-Teil innerhalb eines FileNamens zurück.

Group: Strings 6

3.127 ExtractFilePath

Function *ExtractFilePath (FName : string) : string;*
Liefert den Pfad-Teil innerhalb eines FileNamens zurück.

Group: Strings 6

3.128 F16_BlockRandomWrite

Function *F16_BlockRandomWrite (f: File; pt: pointer; Count : word;* ***var*** *res : Word*) : boolean;
Schreibt einen Datenblock.

Group: FAT16 FileSystem 11

3.129 F16_BlockRead

Function *F16_BlockRead (f : File; pt : pointer; count : word;* ***var*** *res : word) : boolean;*
Liest einen Datenblock.

Group: FAT16 FileSystem 11

3.130 F16_BlockWrite

Function *F16_BlockWrite (f : File; pt : pointer; count: word;* ***var*** *res: word) : boolean;*
Schreibt einen Datenblock.

Group: FAT16 FileSystem

3.131 F16_ChangeDir

Function *F16_ChangeDir (path : TPathStr) : boolean;*
Stellt die DefaultDirectory/Path für alle File Operationen ein.

Group: FAT16 FileSystem

3.132 F16_CheckDisk

Function *F16_CheckDisk : boolean;*
Prüft den Status eines Mediums im Drive.

Group: FAT16 FileSystem

3.133 F16_CheckHandle

Function *F16_CheckHandle (f : File) : tFileAccess;*
Prüft ein FileHandle auf Gültigkeit.

Group: FAT16 FileSystem

3.134 F16_CreateDir

Function *F16_CreateDir (path : TPathStr; DirName : TFileName; aTime, aDate : word): boolean;*
Erstellt eine neue Directory.

Group: FAT16 FileSystem

3.135 F16_DateToStr

Function *F16_DateToStr (FileDate : word) : tF16DateStr;*
Konvertiert ein DOS FileDate in einen String. Format = "yy.mm.dd".

Group: FAT16 FileSystem

3.136 F16_DirGetDate

Function *F16_DirGetDate (path : TPathStr; FDirName : TFileName;* ***var*** *aTime, aDate : word) : boolean;*
Liefert das Erstellungs Datum der Directory.

Group: FAT16 FileSystem

3.137 F16_DiskInit

Function *F16_DiskInit : boolean;*
Initialisiert die angeschlossene Hardware.

Group: FAT16 FileSystem

3.138 F16_DiskReset

Function *F16_DiskReset : boolean;*
Initialisiert alle internen Buffer.

Group: FAT16 FileSystem

3.139 F16_EndOfFile

Function *F16_EndOfFile (f : File) : boolean;*
Gibt ein true zurück, wenn beim Lesen das Dateiende erreicht wurde.

Group: FAT16 FileSystem

3.140 F16_FileAppend

Function *F16_FileAppend (f : File) : boolean**;***
Öffnet ein vorhandenes File zum Schreiben.

Group: FAT16 FileSystem

3.141 F16_FileAssign

Function *F16_FileAssign (**var** f : File; path : TPathStr; fn : TFileName) : boolean;*
Generiert das File (Handle), das für alle weiteren File Operationen benötigt wird.

Group: FAT16 FileSystem

3.142 F16_FileClose

Function *F16_FileClose (**var** f : File) : boolean;*
Gibt ein FileHandle frei.

Group: FAT16 FileSystem

3.143 F16_FileCopy

Function *F16_FileCopy (srcPath : TPathStr; srcFn : TfileName;*
dstPath : TPathStr; dstFn : TFileName) : boolean;
Kopiert eine Datei in ein anderes Verzeichnis/Namen.

Group: FAT16 FileSystem

3.144 F16_FileCreate

Function *F16_FileCreate (Path: TPathStr; FName: TFileName; aAttr : tFAttr;*
aTime, aDate: Word; Size : LongWord) : boolean;
Erstellt eine neue Datei und gibt gleichzeitig auch deren Dateigrösse vor.

Group: FAT16 FileSystem

3.145 F16_FileDelete

Function *F16_FileDelete (path : TPathStr; fn : TFileName) : boolean;*
Löscht eine vorhandene Datei.

Group: FAT16 FileSystem

3.146 F16_FileExist

Function *F16_FileExist (path : TPathStr; fn : TFileName; attr : tFAttr) : boolean;*
Prüft die Existenz einer Datei.

Group: FAT16 FileSystem

3.147 F16_FileGetAttr

Function *F16_FileGetAttr (path : TPathStr; fn : TfileName;* ***var*** *attr : tFAttr) : boolean;*
Liest die Attribute einer vorhandenen Datei.

Group: FAT16 FileSystem

3.148 F16_FileGetDate

Function *F16_FileGetDate (path : TPathStr; fn : TfileName;* ***var*** *aTime, aDate : word) : boolean;*
Liest das File Datum und Uhrzeit einer vorhandenen Datei.

Group: FAT16 FileSystem

3.149 F16_FilePos

Function *F16_FilePos (f : File) : longword****;***
Gibt den aktuellen Lese/Schreib Pointer zurück.

Group: FAT16 FileSystem

3.150 F16_FileRename

Function *F16_FileRename (path : TPathStr; fn, fnNew : TFileName) : boolean;*
Ändert den Filenamen einer vorhandenen Datei.

Group: FAT16 FileSystem

3.151 F16_FileReset

Function *F16_FileReset (f : File) : boolean**;***
Öffnet ein vorhandenes File zum Lesen.

Group: FAT16 FileSystem

3.152 F16_FileRewrite

Function *F16_FileRewrite (f : File**;** attr : tFAttr; aTime, aDate : word) : boolean**;***
Öffnet ein vorhandenes File zum Schreiben.

Group: FAT16 FileSystem

3.153 F16_FileSeek

Function *F16_FileSeek (f : File; p : longword) : longword**;***
Positioniert den Lese/Schreib Pointer.

Group: FAT16 FileSystem

3.154 F16_FileSetAttr

Function *F16_FileSetAttr (path : TPathStr; fn : TFileName; attr : tFAttr) : boolean;*
Verändert die Attribute einer vorhandenen Datei.

Group: FAT16 FileSystem

3.155 F16_FileSetDate

Function *F16_FileSetDate (path : TPathStr; fn : TFileName; aTime, aDate : word) : boolean;*
Verändert das File Datum und Uhrzeit einer vorhandenen Datei.

Group: FAT16 FileSystem

3.156 F16_FileSize

Function *F16_FileSize (path : TPathStr; fn : TFileName; **var** size : longword) : boolean;*
Errechnet die Datei Grösse (Bytes) einer vorhandenen Datei.

Group: FAT16 FileSystem

3.157 F16_FileSizeH

Function *F16_FileSizeH (f: File) : longword;*
Gibt die aktuelle Dateigrösse in Records zurück.

Group: FAT16 FileSystem

3.158 F16_FindFirst

Function *F16_FindFirst (path : TPathStr; fn : TFileName; attr : tFAttr;* ***var*** *sr : TSearchRec) : boolean;*
Eöffnet die Suche nach bestimmten Dateien.

Group: FAT16 FileSystem

3.159 F16_FindNext

Function *F16_FindNext (****var*** *sr : TSearchRec) : boolean;*
Weitersuchen nach dem Eröffnen der Suche mit Funktion FindFirst.

Group: FAT16 FileSystem

3.160 F16_GetCurDir

Function *F16_GetCurDir : TPathStr;*
Gibt das aktuelle DefaultDirectory/Path zurück.

Group: FAT16 FileSystem

3.161 F16_GetDiskError

Function *F16_GetDiskError : tDiskError;*
Liefert den Status der letzten Operation.

Group: FAT16 FileSystem

3.162 F16_GetDiskFree

Function *F16_GetDiskFree : LongWord;*
Gibt den noch freien Speicherplatz in Bytes zurück.

Group: FAT16 FileSystem

3.163 F16_GetDiskSize

Function *F16_GetDiskSize : longword;*
Gibt die Kapazität in Bytes des aktuellen Mediums zurück.

Group: FAT16 FileSystem

3.164 F16_GetUsedHandles

Function *F16_GetUsedHandles : byte;*
Gibt die Anzahl der belegten FileHandles zurück.

Group: FAT16 FileSystem

3.165 F16_PathExist

Function *F16_PathExist (path : TPathStr) : boolean;*
Prüft, ob der angegebene Pfad vorhanden ist.

Group: FAT16 FileSystem

3.166 F16_PathExpand

Function *F16_PathExpand (path : TPathStr;* ***var*** *ExpandedPath : TPathStr) : boolean;*
Expandiert einen relativen Pfad in einen absoluten Pfad.

Group: FAT16 FileSystem

3.167 F16_RandomWrite

Function *F16_RandomWrite (f: File): boolean;*
Öffnet ein vorhandenes File zum Schreiben.

Group: FAT16 FileSystem

3.168 F16_ReadSector

Function *F16_ReadSector (SectNum : longword; pt : Pointer) : Boolean;*
Liest einen physischen 512 Byte Sektor.

Group: FAT16 FileSystem

3.169 F16_RemoveDir

Function *F16_RemoveDir (path : TPathStr; DirName : TFileName) : boolean;*
Löscht eine vorhandene Directory.

Group: FAT16 FileSystem

3.170 F16_StrToDate

Function *F16_StrToDate (strDate : tF16DateStr) : word;*
Konvertiert einen String in ein DOS FileDate word.

Group: FAT16 FileSystem

3.171 F16_StrToTime

Function *F16_StrToTime (strTime : tF16TimeStr) : word;*
Konvertiert einen String in ein DOS FileTime word.

Group: FAT16 FileSystem

3.172 F16_TimeToStr

Function *F16_TimeToStr (FileTime : word) : tF16TimeStr;*
Konvertiert eine DOS FileTime in einen String. Format = "hh.mm".

Group: FAT16 FileSystem 11

3.173 F16_WriteSector

Function *F16_WriteSector (SectNum : longword; pt : Pointer) : Boolean;*
Schreibt einen physischen 512 Byte Sektor.

Group: FAT16 FileSystem 11

3.174 FileAppend

Function *FileAppend (**const** f : file; **const** Buf[; **const** Count: word]) : word;*
Hängt Records ans Ende des Files an.

Group: 8bit FileSystem 7

3.175 FileChangeDir

Function *FileChangeDir (**const** fn : tFName; **const** dir : char) : boolean;*
Verschiebt eine Datei in ein anderes Directory.

Group: 8bit FileSystem 7

3.176 FileClose

Function *FileClose (**var** f : file) : boolean;*
Schliesst die geöffnete Datei.

Group: 8bit FileSystem 7

3.177 FileCreate

Function *FileCreate (**const** fn : tFName) : boolean;*
Erstellt eine neue leere Datei.

Group: 8bit FileSystem 7

3.178 FileDelete

Function *FileDelete (**const** fn : tFName) : boolean;*
Löscht die Datei *fn*.

Group: 8bit FileSystem 7

3.179 FileExists

Function *FileExists (**const** fn : tFName) : boolean;*
Stellt fest ob die Datei *fn* existiert.

Group: 8bit FileSystem

3.180 FileFirst

Function *FileFirst (**var** st : tFName; **const** fn : tFName) : boolean;*
Eröffnet die Datei Suchfunktion.

Group: 8bit FileSystem

3.181 FileGetAttr

Function *FileGetAttr (**const** fn : tFName) : TFileAttributes;*
Liest alle Attribute dieser Datei.

Group: 8bit FileSystem

3.182 FileHandleCheck

Function *FileHandleCheck (**const** f : file) : boolean**;***
Prüft eine FileHandle Variable auf Gültigkeit.

Group: 8bit FileSystem

3.183 FileNext

Function *FileNext (**var** st : tFName) : boolean;*
Setzt das Suchen nach einer *FileFirst* Funktion fort.

Group: 8bit FileSystem

3.184 FileOpen

Function *FileOpen (**var** f : file; **const** fn : tFName) : boolean;*
Öffnet eine vorhandene Datei zum Lesen oder Schreiben.

Group: 8bit FileSystem

3.185 FilePos

Function *FilePos (**const** f : file) : longword**;***
Gibt die aktuelle Position des Schreib/Lesepointers der Datei zurück.

Group: 8bit FileSystem

3.186 FileRead

Function *FileRead (**const** f : file; **var** Buf [; **const** Count: word]) : word;*
Liest ab dem aktuellen ReadWrite Pointer die Anzahl *Count* Records.

Group: 8bit FileSystem

3.187 FileRename

Function *FileRename (**const** fn, fnNew : tFName) : boolean;*
Benennt eine Datei um.

Group: 8bit FileSystem

3.188 FileReset

Function *FileReset (**const** f : file) : boolean**;***
Setzt den ReadWrite Pointer für diese Datei an den Datei Anfang.

Group: 8bit FileSystem

3.189 FileRewrite

Function *FileRewrite (**const** f : file) : boolean**;***
Schliesst die geöffnete Datei, löscht diese und eröffnet eine neue leere Datei.

Group: 8bit FileSystem

3.190 FileSeek

Function *FileSeek (**const** f : file; **const** p : longword) : longword;*
Positioniert den Lese/Schreibpointer.

Group: 8bit FileSystem

3.191 FileSetAttr

Function *FileSetAttr (**const** fn : tFName; attr : TFileAttributes) : boolean;*
Ändert die Attribute dieser Datei.

Group: 8bit FileSystem

3.192 FileSize

Function *FileSize (**const** fn : tFName [, f : fileType|type]) : longword;*
Errechnet die Dateigrösse des Files *fn* in Records.

Group: 8bit FileSystem

3.193 FileSysReset

Procedure *FileSysReset;*
Schliesst alle internen Buffer.

Group: 8bit FileSystem

3.194 FileWrite

Function *FileWrite (**const** f : file; **const** Buf [; **const** Count: word]) : word;*
Schreibt ab dem aktuellen ReadWrite Pointer die Anzahl *Count* Records.

Group: 8bit FileSystem

3.195 FillBlock

Procedure *FillBlock (start, size : word; fill : byte);*
Füllt bzw. löscht einen Speicherbereich mit einem Byte oder Char.

Group: System

3.196 FillRandom

Procedure *FillRandom (p : pointer; cnt : word);*
füllt einen Speicher Bereich mit Zufalls Werten.

Random muss importiert werden:
From *System* ***Import*** *Random****;***

Group: System

3.197 Fix64ArcCos

Function *Fix64ArcCos (const cosine: fix64): fix64;*
liefert den Winkel im Bogenmaß für einen vorgegebenen Cosinus diese Winkels.
Siehe auch Fix64ArcCosD

Group: Fix64

3.198 Fix64ArcCosD

Function *Fix64ArcCosD (const cosine: fix64): fix64;*
liefert den Winkel in Grad für einen vorgegebenen Cosinus diese Winkels.
Siehe auch Fix64ArcCos

Group: Fix64

3.199 Fix64ArcCosh

Function *Fix64ArcCosh (const a: fix64): fix64;*
liefert den Winkel im Bogenmaß für einen vorgegebenen hyperbolischen Cosinus diese Winkels.

Group: Fix64

3.200 Fix64ArcCot

Function *Fix64ArcCot (const cotangent: fix64): fix64;*
liefert den Winkel im Bogenmaß für einen vorgegebenen Cotangens diese Winkels.
Siehe auch Fix64ArcCotD

Group: Fix64

3.201 Fix64ArcCotD

Function *Fix64ArcCotD (const cotangent: fix64): fix64;*
liefert den Winkel in Grad für einen vorgegebenen Cotangens diese Winkels.
Siehe auch Fix64ArcCot

Group: Fix64

3.202 Fix64ArcCsc

Function *Fix64ArcCsc (const cosecant: fix64): fix64;*
liefert den Winkel im Bogenmaß für einen vorgegebenen Cosecans diese Winkels.
Siehe auch Fix64ArcCscD

Group: Fix64

3.203 Fix64ArcCscD

Function *Fix64ArcCscD (const cosecant: fix64): fix64;*
liefert den Winkel in Grad für einen vorgegebenen Cosecans diese Winkels.
Siehe auch Fix64ArcCsc

Group: Fix64

3.204 Fix64ArcSec

Function *Fix64ArcSec (const secant: fix64): fix64;*
liefert den Winkel im Bogenmaß für einen vorgegebenen Secans diese Winkels.
Siehe auch Fix64ArcSecD

Group: Fix64

3.205 Fix64ArcSecD

Function *Fix64ArcSecD (const secant: fix64): fix64;*
liefert den Winkel in Grad für einen vorgegebenen Secans diese Winkels.
Siehe auch Fix64ArcSec

Group: Fix64

3.206 Fix64ArcSin

Function *Fix64ArcSin (const sine: fix64): fix64;*
liefert den Winkel im Bogenmaß für einen vorgegebenen Sinus diese Winkels.
Siehe auch Fix64ArcSinD

Group: Fix64

3.207 Fix64ArcSinD

Function *Fix64ArcSinD (const sine: fix64): fix64;*
liefert den Winkel in Grad für einen vorgegebenen Sinus diese Winkels.
Siehe auch Fix64ArcSin

Group: Fix64

3.208 Fix64ArcSinh

Function *Fix64ArcSinh (const a: fix64): fix64;*
liefert den Winkel im Bogenmaß für einen vorgegebenen hyperbolischen Sinus diese Winkels.

Group: Fix64

3.209 Fix64ArcTan

Function *Fix64ArcTan (const tangent: fix64): fix64;*
liefert den Winkel im Bogenmaß für einen vorgegebenen Tangens diese Winkels.
siehe auch Fix64ArcTanD, Fix64ArcTan2, Fix64ArcTan2D

Group: Fix64

3.210 Fix64ArcTan2

Function *Fix64ArcTan2 (const y, x: fix64): fix64;*
Abwandlung der Arctangens Funktion: für beliebige reelle Argumente x und y, die nicht beide Null sind,
liefert arctan2(x,y) den Winkel im Bogenmaß zw. der positiven x-Achse der Ebene und dem durch die
Koordinaten (x,y) gegebenen Punkt darauf.
siehe auch Fix64ArcTan2D, Fix64ArcTan, Fix64ArcTanD

Group: Fix64

3.211 Fix64ArcTan2D

Function *Fix64ArcTan2D (const y, x: fix64): fix64;*
Abwandlung der Arctangens Funktion: für beliebige reelle Argumente x und y, die nicht beide Null sind,
liefert arctan2(x,y) den Winkel in Grad zw. der positiven x-Achse der Ebene und dem durch die Koordinaten (x,y) gegebenen Punkt darauf.
siehe auch Fix64ArcTanD, Fix64ArcTan, Fix64ArcTan2

Group: Fix64

3.212 Fix64ArcTanD

Function *Fix64ArcTanD (const tangent: fix64): fix64;*
liefert den Winkel in Grad für einen vorgegebenen Tangens diese Winkels.
siehe auch Fix64ArcTan2D, Fix64ArcTan, Fix64ArcTan2

Group: Fix64

3.213 Fix64ArcTanh

Function *Fix64ArcTanh (const a: fix64): fix64;*
liefert den Winkel im Bogenmaß für einen vorgegebenen hyperbolischen Tangens diese Winkels.

Group: Fix64

3.214 Fix64Cos

Function *Fix64Cos (const degrees: fix64): fix64;*
liefert den Cosinus des Arguments (Argument=Winkel im Bogenmass).
Siehe auch Fix64CosD

Group: Fix64

3.215 Fix64CosD

Function *Fix64CosD (const degrees: fix64): fix64;*
liefert den Cosinus des Arguments (Argument=Winkel in Grad).
Siehe auch Fix64Cos

Group: Fix64

3.216 Fix64Cosh

Function *Fix64Cosh (const a: fix64): fix64;*
liefert den Cosinus Hyperbolicus von a.

Group: Fix64

3.217 Fix64Cot

Function *Fix64Cot (const radians: fix64): fix64;*
liefert den Cotangens des Arguments (Argument=Winkel im Bogenmaß).
Siehe auch Fix64CotD

Group: Fix64

3.218 Fix64CotD

Function *Fix64CotD (const degrees: fix64): fix64;*
liefert den Cotangens des Arguments (Argument=Winkel in Grad).
Siehe auch Fix64Cot

Group: Fix64

3.219 Fix64Csc

Function *Fix64Csc (const radians: fix64): fix64;*
liefert den Cosecans des Arguments (Argument=Winkel im Bogenmaß).
Siehe auch Fix64CscD

Group: Fix64

3.220 Fix64CscD

Function *Fix64CscD (const degrees: fix64): fix64;*
liefert den Cosecans des Arguments (Argument=Winkel in Grad).
Siehe auch Fix64Csc

Group: Fix64

3.221 Fix64DegToRad

Function *Fix64DegToRad (const degrees: fix64): fix64;*
liefert das Bogenmaß Arguments (Argument im Bogenmass).
Siehe auch Fix64DegToRadD

Group: Fix64

3.222 Fix64DegToRadD

Function *Fix64DegToRadD (const degrees: fix64): fix64;*
liefert das Bogenmaß Arguments (Argument in Grad).
Siehe auch Fix64DegToRad

Group: Fix64

3.223 Fix64DivInt

Function *Fix64DivInt (const a: fix64; const b: integer): fix64;*
liefert den Quotient aus Fix64 dividiert durch Integer.
Siehe auch Fix64DivLong

Group: Fix64

3.224 Fix64DivLong

Function *Fix64DivLong (const a: fix64; const b: longint): fix64;*
liefert den Quotient aus Fix64 dividiert durch Long Integer.
Siehe auch Fix64DivInt

Group: Fix64

3.225 Fix64Even

Function *Fix64Even (const a: fix64): boolean;*
liefert *true* wenn das Argument gerade ist.
Siehe auch Fix64Odd

Group: Fix64

3.226 Fix64Exp

Function *Fix64Exp (const a: fix64): fix64;*
liefert "e" hoch "a". "*e*" ist die Basis des natürlichen Logarithmus (2.71828...).

Group: Fix64

3.227 Fix64Integrate

Function *Fix64Integrate (const aold, anew: fix64; const factor: byte): fix64;*
Integriert einen neuen Wert zu einem schon vorhandenen.
Berechnung: *result:= ((aold * factor) + anew)* ***div*** *(fact + 1);*
Ein Überflauf in der Multiplikation kann hier auftreten! Vorsicht mit grossen Zahlen !

Group: Fix64

3.228 Fix64IsPowOfTwo

Function *Fix64IsPowOfTwo (const a: fix64): boolean;*
liefert *true* wenn "a" eine 2'er Potenz ist.

Group: Fix64

3.229 Fix64Ln

Function *Fix64Ln (const a: fix64): fix64;*
liefert den natürlichen Logarithmus von a zur Basis e.
"*e*" ist die Basis des natürlichen Logarithmus (2.71828...).
Ist identisch zu Fix64LogN. Siehe auch Fix64Log und Fix64Log10

Group: Fix64

3.230 Fix64Log

Function *Fix64Log (const a, base: fix64): fix64;*
liefert den Logarithmus von a zur Basis "*base*".
Siehe auch Fix64Log10 und Fix64Ln

Group: Fix64

3.231 Fix64Log10

Function *Fix64Log10 (const a: fix64): fix64;*
liefert den Logarithmus von a zur Basis 10.
Siehe auch Fix64Log und Fix64Ln

Group: Fix64

3.232 Fix64LogN

Function *Fix64LogN (const a: fix64): fix64;*
liefert den natürlichen Logarithmus von a zur Basis e.
"*e*" ist die Basis des natürlichen Logarithmus (2.71828...).
Ist identisch zu Fix64Ln. Siehe auch Fix64Log und Fix64Log10

Group: Fix64

3.233 Fix64Mod

Function *Fix64Mod (const a, modulus: fix64): fix64;*
liefert "a mod *modulus*". Beide Argumente sind vom Typ Fix64.
siehe auch: Fix64ModInt

Group: Fix64

3.234 Fix64ModInt

Function *Fix64ModInt (const a: fix64; modulus: integer): fix64;*
liefert "a mod *modulus*". *modulus* ist hier vom Typ Integer.
siehe auch: Fix64Mod

Group: Fix64

3.235 Fix64MulInt

Function *Fix64MulInt(const a: fix64; const b: integer): fix64;*
liefert das Produkt eines Fix64 und eines Integer.
siehe auch: Fix64MulLong

Group: Fix64

3.236 Fix64MulLong

Function *Fix64MulLong(const a: fix64; const b: longint): fix64;*
liefert das Produkt eines Fix64 und eines Long Integer.
siehe auch: Fix64MulInt

Group: Fix64

3.237 Fix64Odd

Function *Fix64Odd(const a: fix64): boolean;*
liefert *true* wenn das Argument ungerade ist.
Siehe auch Fix64Even

Group: Fix64

3.238 Fix64Power

Function *Fix64Power (const base, exponent: fix64): fix64;*
liefert *base* ^ *exponent*. "*base*" und "*exponent*" sind vom Typ Fix64.
siehe auch: Fix64PowerInt

Group: Fix64

3.239 Fix64PowerInt

Function *Fix64PowerInt (const base: fix64; const exponent: integer): fix64;*
liefert *base* ^ *exponent*. "*base*" ist vom vom Typ Fix64, "*base*" vom Typ Integer.
Siehe auch: Fix64Power

Group: Fix64

3.240 Fix64Quadrant

Function *Fix64Quadrant (const radians: fix64): byte;*
liefert den Quadrant (1..4) des Winkels (Winkel im Bogenmaß).

Group: Fix64

3.241 Fix64RadToDeg

Function *Fix64RadToDeg (const radians: fix64): fix64;*
liefert das Gradmaß des Arguments (Argument im Bogenmaß).
Siehe auch Fix64DegToRad

Group: Fix64

3.242 Fix64Sec

Function *Fix64Sec (const radians: fix64): fix64;*
liefert den Secans des Arguments (Argument=Winkel im Bogenmaß).
Siehe auch Fix64SecD

Group: Fix64

3.243 Fix64SecD

Function *Fix64SecD (const degrees: fix64): fix64;*
liefert den Secans des Arguments (Argument=Winkel in Grad).
Siehe auch Fix64Sec

Group: Fix64

3.244 Fix64Sin

Function *Fix64Sin (const radians: fix64): fix64;*
liefert den Sinus des Arguments (Argument=Winkel im Bogenmaß).
Siehe auch Fix64SinD

Group: Fix64

3.245 Fix64SinD

Function *Fix64SinD (const degrees: fix64): fix64;*
liefert den Sinus des Arguments (Argument=Winkel in Grad).
Siehe auch Fix64Sin

Group: Fix64

3.246 Fix64Sinh

Function *Fix64Sinh (const a: fix64): fix64;*
liefert den Sinus Hyperbolicus von a.

Group: Fix64

3.247 Fix64Sqrt

Function *Fix64Sqrt (f : fix64) : fix64;*
Liefert die Quadratwurzel des Fix64-Arguments.

Bitte beachten:
Fix64Sqrt hat einen Frame Bedarf von 64 Bytes!

Info:
Fix64Sqrt braucht 2.2msec @16MHz, Genauigkeit: 9 Nachkomma Stellen
(nur in der AVRco Profi Version verfügbar)
Sqrt (fix64) braucht 600usec @16MHz, Genauigkeit: 5 Nachkomma Stellen

Group: Fix64

3.248 Fix64Tan

Function *Fix64Tan (const radians: fix64): fix64;*
liefert den Tangens des Arguments (Argument=Winkel im Bogenmaß).
Siehe auch Fix64TanD

Group: Fix64

3.249 Fix64TanD

Function *Fix64TanD (const degrees: fix64): fix64;*
liefert den Tangens des Arguments (Argument=Winkel in Grad).
Siehe auch Fix64Tan

Group: Fix64

3.250 Fix64Tanh

Function *Fix64Tanh (const a: fix64): fix64;*
liefert den Tangens Hyperbolicus von a.

Group: Fix64

3.251 Fix64ToFloat

Function *Fix64ToFloat (const a: fix64): float;*
Type Cast: konvertiert einen Fix64 in einen Float.

Group: Fix64

3.252 Fix64ToHex

Function *Fix64ToHex (**const** f : Fix64) : string;*
Konvertiert einen Fix64 Wert in einen hex-String.

Groups: Strings, Fix64

3.253 Fix64ToInt

Function *Fix64ToInt (const a: fix64): integer;*
Type Cast: konvertiert einen Fix64 in einen Integer.

Group: Fix64 4

3.254 Fix64ToLongInt

Function *Fix64ToLongInt (const a: fix64): longint;*
Type Cast: konvertiert einen Fix64 in einen Long Integer.

Group: Fix64 4

3.255 Fix64ToStr

Function *Fix64ToStr(f : Fix64[:int : byte[:frac : char] [:space : char]]) : string;*
Konvertiert einen Fix64 Wert in einen String.
Die optionalen Parameter int, frac und space formatieren den String.

Groups: Strings 6, Fix64 4

3.256 Fix64ValueInTolerance

Function *Fix64ValueInTolerance (const a, aref, atol: fix64): boolean;*
vergleicht den Inhalt von „a" mit der Grenze "vmin", die aus (aref - atol) gebildet wird und "vmax", die aus (aref + atol) gebildet wird.
Überschreitet der Wert von value eine der Grenzen, so wird das Ergebnis false, ansonsten true.
Siehe auch Fix64ValueInToleranceP

Group: Fix64 4

3.257 Fix64ValueInToleranceP

Function *Fix64ValueInToleranceP (const a, aref, atol: fix64): boolean;*
vergleicht den Inhalt von "a" mit der Grenze "vmin", die aus (aref - (aref div 100) * atol) und "vmax", die aus (aref + (aref div 100) * tol) gebildet wird.
Überschreitet der Wert von value eine der Grenzen, so wird das Ergebnis false, ansonsten true.
Der Wert von "atol" muss im Bereich 0..100 liegen, da es sich um einen Prozentsatz handelt.
Diese Funktion ist identisch zu Fix64ValueInTolerance. Nur dass hier die Toleranz nicht absolut, sondern relativ in Prozent angegeben wird.

Group: Fix64 4

3.258 FlashClearPage

Procedure *FlashClearPage;*
Löscht den temporären CPU Buffer mit $FF.

Group: Flash Writer 11

3.259 FlashCopyF2R

Procedure *FlashCopyF2R (p : pointer);*
Kopiert die Flash Page, auf die *FLASH_ADDR* zeigt, in den Buffer im RAM.

Group: Flash Writer 11

3.260 FlashCopyR2F

Procedure *FlashCopyR2F (p : pointer);*
Kopiert die zuvor mit *FlashCopyF2R* gerettete Flash Page in den temporären Buffer der CPU.

Group: Flash Writer 11

3.261 FlashDownLoader

Procedure *FlashDownLoader;*
Aufruf des Download Monitors.

Group: Diverse 4

3.262 FlashErasePage

Procedure *FlashErasePage;*
Löscht die mit *FLASH_ADDR* bestimmte Page im Flash.

Group: Flash Writer 11

3.263 FlashInitPage

Procedure *FlashInitPage (**const** addr : LongWord);*
Muss vor jeder neuen Page aufgerufen werden.

Group: Flash Writer 11

3.264 FlashLoaderExit

Procedure *FlashLoaderExit;*
Callback Funktion des Flash Downloaders.

Group: Diverse 4

3.265 FlashLoaderInit

Procedure *FlashLoaderInit;*
Callback Funktion des Flash Downloaders.

Group: Diverse 4

3.266 FlashLoaderRecv

Procedure *FlashLoaderRecv;*
Callback Funktion des Flash Downloaders.

Group: Diverse 4

3.267 FlashLoaderTransm

Procedure *FlashLoaderTransm;*
Callback Funktion des Flash Downloaders.

Group: Diverse 4

3.268 FlashProgPage

Procedure *FlashProgPage;*
Der Temporäre Buffer der CPU wird in das Flash programmiert.

Group: Flash Writer 11

3.269 FlashPtr

Function *FlashPtr (p:pointer): pointer;*
Lässt den Pointer ins Flash zeigen.

Group: System 7

3.270 FlashReadFuses

Function *FlashReadFuses (FuseGroup : byte) : byte;*
Die Fusebits können zur Laufzeit ausgelesen werden.

Group: Flash Writer 11

3.271 FlashReadPage

Procedure *FlashReadPage;*
Liest die Flashpage, auf die *FLASH_ADDR* zeigt, aus dem ROM.

Group: Flash Writer 11

3.272 FlashWriteFuses

Procedure *FlashWriteFuses (FuseGroup, fsBits : byte);*
Die Fusebits können zur Laufzeit geschrieben werden.

Group: Flash Writer 11

3.273 FlashWritePage

Procedure *FlashWritePage (**const** parm : word);*
Schreibt ein Wort in den temporären Buffer der CPU.

Group: Flash Writer 11

3.274 FloatAsLong

Function *FloatAsLong (f : float) : longword;*
Wandelt das Argument in ein LongWord.
Führt jedoch **keine Typ Konvertierung** durch!

Group: Diverse 4

3.275 FloatToFix64

Function *FloatToFix64 (const a: float): fix64;*
Type Cast: konvertiert einen Float in einen Fix64.

Group: Fix64 4

3.276 FloatToStr

Function *FloatToStr (f : float) : string;*
Konvertiert Floating Point Wert in einen String.

Group: Strings 6

3.277 FlushBuffer

Procedure *FlushBuffer (Buffer : tBuffer);*

```
FlushBuffer (RxBuffer1);
FlushBuffer (RxBuffer2);
. . .
FlushBuffer (TxBuffer1);
FlushBuffer (TxBuffer2);
. . .
```

XMega:

```
FlushBuffer (RxBufferC0);
FlushBuffer (RxBufferC1);
FlushBuffer (RxBufferD0);
. . .
FlushBuffer (TxBufferC0);
FlushBuffer (TxBufferC1);
FlushBuffer (TxBufferD0);
. . .
```

Löscht den Inhalt eines Buffers der seriellen Schnittstellen.

Group: SerPorts 30

3.278 Frac

Function *Frac (f : float|fix64) : float|fix64;*
Liefert den Bruchanteil des Arguments zurück.

Group: Maths 5

3.279 FreeMem

Function *FreeMem (**var** ptr : pointer) : boolean;*
Gibt Speicher an den Heap zurück.

Group: Diverse 4

3.280 FreqCountRestart

Procedure *FreqCountRestart;*
Setzt den Frequenzzähler zurück.

Group: Frequency Counter/Timer 11

3.281 FreqCountRestart2

Procedure *FreqCountRestart2;*
Setzt den Frequenzzähler zurück.

Group: Frequency Counter/Timer 11

3.282 gClearPixel

Procedure *gClearPixel (Px, Py : integer);*
Löscht einen Pixel an der Stelle „Px,Py" im aktuellen ViewPort.

Group: LCD Graphic 22

3.283 gClearView

Procedure *gClearView (ClearMode : TWriteMode);*
Löscht das aktuelle ViewPort.

Group: LCD Graphic 22

3.284 gClrScr

Procedure *gClrScr (pattern : byte);*
Löscht das komplette Display mit dem Byte „pattern".

Group: LCD Graphic 22

3.285 gDispRefresh

Procedure *gDispRefresh;*
Schreibt den kompletten internen Refresh Buffer in das Display.

Group: LCD Graphic 22

3.286 gDrawBitMap

Procedure *gDrawBitMap (Xs, Ys : integer; source : pointer; DrawMode : TWriteMode);*
Kopiert ein BitMap aus dem RAM/ROM in das aktuelle ViewPort.

Group: LCD Graphic 22

3.287 gDrawBitMapN

Procedure *gDrawBitMapN (Xs, Ys : integer; source : pointer; DrawMode : TWriteMode);*
Kopiert ein BitMap aus dem RAM/ROM in das aktuelle ViewPort.

Group: LCD Graphic 22

3.288 gDrawCircle

Procedure *gDrawCircle (Xc, Yc, R : integer; pattern : byte);*
Zeichnet einen Kreis in den aktuellen ViewPort.

Group: LCD Graphic 22

3.289 gDrawLine

Procedure *gDrawLine (Xs, Ys, Xe, Ye : integer; pattern : byte);*
Zeichnet eine Linie in den aktuellen ViewPort.

Group: LCD Graphic 22

3.290 gDrawLineTo

Procedure *gDrawLineTo (Xd, Yd : integer; pattern : byte);*
Zeichnet eine Linie in den aktuellen ViewPort.

Group: LCD Graphic 22

3.291 gDrawLineToRel

Procedure *gDrawLineToRel (Xr, Yr : integer; pattern : byte);*
Zeichnet eine Linie in den aktuellen ViewPort.

Group: LCD Graphic 22

3.292 gDrawRect

Procedure *gDrawRect (Xs, Ys, Xe, Ye : integer; pattern : byte);*
Zeichnet ein Rechteck in den aktuellen ViewPort.

Group: LCD Graphic 22

3.293 gDrawString

Procedure *gDrawString (X,Y: integer; zx,zy: byte; rot: TTxtRotate; str: TGraphString);*
Zeichnet den String „str" an die logische ViewPort Position „X,Y".

Group: LCD Graphic 22

3.294 gDrawStringRel

Procedure *gDrawStringRel (zx, zy : byte; rot : TTxtRotate; str : TGraphString);*
Zeichnet den String „str" an die relative ViewPort Position „X,Y".

Group: LCD Graphic 22

3.295 GetAdc

Function *GetADC (chan : byte) : word;*

Achtung: der AVRco zählt die Kanäle 1, 2, 3, ... (entsprechend 0, 1, 2, ... beim Controller)

bzw.

Function *GetADC : word;*
wenn nur ein einziger Kanal definiert ist

Liefert Daten des ADC's. Funktion zum Auslesen der ADC Kanäle.

Group: ADC 8

3.296 GetAVfilter

Function *GetAVfilter (**var** Filter : AVfilter) : type;*
Berechnet den aktuellen Mittelwert ohne den Filterinhalt zu ändern.

Group: Diverse 4

3.297 GetBankNum

Function *GetBankNum (bankedVar): byte;*
Liefert die Bank Nummer der gebankten Variablen „bankedVar" zurück.

Group: Banking Port 9

3.298 GetCurDir

Function *GetCurDir : char; // '0'..'9'*
Gibt das aktuelle *DefaultDirectory* zurück.

Group: 8bit FileSystem 7

3.299 GetCurDisk

Function *GetCurDisk : char; // 'A'..'D'*
Gibt das aktuelle *DefaultDrive* zurück.

Group: 8bit FileSystem 7

3.300 GetCurProcess

Function *GetCurProcess : byte;*
Liefert die Prozess ID des aktuellen Prozesses/Task.

Group: MultiTasking 5

3.301 GetExceptResult

Function *GetExceptResult : byte;*
Parameter *d*er Exception. Siehe try

Group: System 7

3.302 GetFrameFree

Function *GetFrameFree (p : Process) : word;*
Ermittelt den Frame Verbrauch von Prozessen zur Laufzeit.

Group: System 7

3.303 GetFreqCounter

Function *GetFreqCounter: word;*
Liefert das Ergebnis der letzten Frequenz Messung zurück.

Group: Frequency Counter/Timer 11

3.304 GetFreqCounter2

Function *GetFreqCounter2: word;*
Liefert das Ergebnis der letzten Frequenz Messung zurück.

Group: Frequency Counter/Timer 11

3.305 GetFreqCounter2L

Function *GetFreqCounter2L: longword;*
Liefert das Ergebnis der letzten Frequenz Messung zurück.

Group: Frequency Counter/Timer 11

3.306 GetFreqCounterL

Function *GetFreqCounterL: longword;*
Liefert das Ergebnis der letzten Frequenz Messung zurück.

Group: Frequency Counter/Timer 11

3.307 GetFreqCountMode

Function *GetFreqCountMode : tFreqCountMode;*
Liefert als Ergebnis den aktuellen Modus des Zählers zurück.

Group: Frequency Counter/Timer 11

3.308 GetFreqCountMode2

Function *GetFreqCountMode2 : tFreqCountMode;*
Liefert als Ergebnis den aktuellen Modus des Zählers zurück.

Group: Frequency Counter/Timer 11

3.309 GetFreqCountOvrFlow

Function *GetFreqCountOvrFlow: byte;*
Liefert als Ergebnis einen evtl. vorhandenen Überlauf.

Group: Frequency Counter/Timer 11

3.310 GetFreqCountOvrFlow2

Function *GetFreqCountOvrFlow2: byte;*
Liefert als Ergebnis einen evtl. vorhandenen Überlauf.

Group: Frequency Counter/Timer 11

3.311 GetIncrementRel

Function *GetIncrementRel : integer [longint];*
Liefert den aktuellen internen relativen Zählerstand.

Group: Increment Counter 15

3.312 GetIncrementVal

Function *GetIncrementVal : integer [longint];*
Liefert den aktuellen internen Zählerstand.

Group: Increment Counter 15

3.313 GetIncrRel4

Function *GetIncrRel4 (chan : byte) : integer [longint];*
Liefert den aktuellen reativen internen Zählerstand von *chan.*

Group: Increment Counter 4chan 15

3.314 GetIncrVal4

Function *GetIncrVal4 (chan : byte) : integer [longint];*
Liefert den aktuellen internen Zählerstand von *chan.*

Group: Increment Counter 4chan 15

3.315 GetKey

Function *GetKey : Keys;*
Gibt die erste gefundene aktive Taste zurück.

Group: KeyBoard 4x4 16

3.316 GetKey8

Function *GetKey8 : Keys;*
Gibt die erste gefundene aktive Taste zurück.

Group: KeyBoard 8x8 17

3.317 GetKeyRaised

Function *GetKeyRaised : Keys;*
Gibt die erste gefundene aktive, gelatchte Taste zurück.

Group: KeyBoard 4x4 16

3.318 GetKeyRaised8

Function *GetKeyRaised8 : Keys;*
Gibt die erste gefundene aktive, gelatchte Taste zurück.

Group: KeyBoard 8x8 17

3.319 GetLargestBlock

Function *GetLargestBlock: word;*
Gibt die Grösse des grössten, zusammenhängenden freien Speicherblocks zurück.

Group: Diverse 4

3.320 GetMem

Function *GetMem (**var** ptr : pointer [; **const** size : word]) : boolean;*
Fordert Speicher vom Heap an.

Group: Diverse 4

3.321 GetMemAvail

Function *GetMemAvail : word;*
Gibt die Summe des freien Speichers zurück.

Group: Diverse 4

3.322 GetPeriority

Function *GetPriority (prcs : process|task) : byte;*
Liefert die aktuelle Priorität eines Prozess/Task.

Group: MultiTasking 5

3.323 GetProcessID

Function *GetProcessID (ProcName) : byte;*
Die Prozess ID eines Prozesses/Task wird abgefragt.

Group: MultiTasking 5

3.324 GetProcessState

Function *GetProcessState (name : process|Task) : tProcessState;*
Der Status eines Prozesses oder Tasks wird abgefragt.

Group: MultiTasking 5

3.325 GetPulseCount

Function *GetPulseCount : longword;*
Liefert den aktuellen internen Zählerstand.

Group: Pulse Counter 27

3.326 GetPulseCount2

Function *GetPulseCount2 : longword;*
Liefert den aktuellen internen Zählerstand.

Group: Pulse Counter 27

3.327 GetStackFree

Function *GetStackFree (p : Process) : word;*
Ermittelt den Stack Verbrauch von Prozessen zur Laufzeit.

Group: System 7

3.328 GetSysTimer

Function *GetSysTimer (tm : tSysTimer) : byte|word;*
Gibt den aktuellen Wert eines SysTimers zurück.

Group: System 7

3.329 GetTable

Function *GetTable (t : Table; index : byte) : type;*
Liefert ein Mitglied einer LookUp-Table zurück.

Group: System 7

3.330 GetTaskFrameFree

Function *GetTaskFrameFree : word;*
Die Funktion liefert ein Wort mit der Anzahl von Bytes, die im Frame Stack noch unbenutzt sind.

Group: System 7

3.331 GetTaskStackFree

Function *GetTaskStackFree : word;*
Ermittelt den Stack Verbrauch der Tasks zur Laufzeit.
Die Funktion liefert ein Wort mit der Anzahl von Bytes, die im Task Stack noch unbenutzt sind.

Group: System 7

3.332 GetTimeCounter

Function *GetTimeCounter: word;*
Liefert das Ergebnis der letzten Zeit Messung zurück.

Group: Frequency Counter/Timer 11

3.333 GetTimeCounter2

Function *GetTimeCounter2 : word;*
Liefert das Ergebnis der letzten Zeit Messung zurück.

Group: Frequency Counter/Timer 11

3.334 GetTimeCounterP

Function *GetTimeCounterP (**var** Count1, Count2: word) : boolean;*
Liefert als Ergebnis die Counter Werte der letzten Zeitmessung.
Count1: high Periode, Count2: low Periode. False bei Overflow.

Group: Frequency Counter/Timer 11

3.335 GetTimeCounterP2

Function *GetTimeCounterP2 (**var** Count1, Count2: word) : boolean;*
Liefert als Ergebnis die Counter Werte der letzten Zeitmessung.
Count1: high Periode, Count2: low Periode. False bei Overflow.

Group: Frequency Counter/Timer 11

3.336 GetTWIslaveStat

Function *GetTWIslaveStat (node : byte) : tTWINetState;*
Stell den Status der lokalen Rx und Tx Buffer bzw. Frames fest.

Group: TWI Network 44

3.337 GetWatchDogFlag

Function *GetWatchDogFlag : byte;*
Liefert ein Byte mit einer Kopie des "MCUSR" Register Inhaltes.

Group: System 7

3.338 gFillCircle

Procedure *gFillCircle (Xc, Yc, R : integer; pattern : byte);*
Füllt einen Kreis in den aktuellen ViewPort.

Group: LCD Graphic 22

3.339 gFillRect

Procedure *gFillRect (Xs, Ys, Xe, Ye : integer; pattern : byte);*
Füllt ein Rechteck in den aktuellen ViewPort.

Group: LCD Graphic 22

3.340 gFrameView

Procedure *gFrameView (ViewPort: byte);*
Zeichnet einen Rahmen um das ViewPort.

Group: LCD Graphic 22

3.341 gGetCurView

Function *gGetCurView : byte;*
Liefert als Ergebnis das aktuelle ViewPort.

Group: LCD Graphic 22

3.342 gGetLineColor

Function *gGetLineColor : byte;*
Liefert die Farbe für das aktuelle ViewPort.

Für den Treiber muss Farbe importiert werden:
From *LCDGraphic* ***Import*** *GraphColor;*

Group: LCD Graphic 22

3.343 gGetLineMode

Function *gGetLineMode : TWriteMode;*
Liefert den Schreib-Modus des aktuellen ViewPorts zurück.

Group: LCD Graphic 22

3.344 gGetTextBkGnd

Function *gGetTextBkGnd : TTextBkGnd;*
Liefert den aktuellen Text-Hintergrund Mode für das aktuelle ViewPort.

Group: LCD Graphic 22

3.345 gGetTextColor

Function *gGetTextColor : byte;*
Liefert die Farbe für das aktuelle ViewPort.

Für den Treiber muss Farbe importiert werden:
From *LCDGraphic* ***Import*** *GraphColor;*

Group: LCD Graphic 22

3.346 gGetTextJustify

Function *gGetTextJustify (**var** Horiz : TtxtAlHor; **var** Vert : TTxtAlVert);*
Liefert die aktuelle Text Alignment Einstellung für das aktuelle ViewPort.

Group: LCD Graphic 22

3.347 gGetTextMode

Function *gGetTextMode : TWriteMode;*
Liefert den aktuellen Text Schreibmodus für das aktuelle ViewPort.

Group: LCD Graphic 22

3.348 gMoveTo

Procedure *gMoveTo (Xd, Yd : integer);*
Setzt den virtuellen Zeichen Cursor auf die Koordinate „Xd, Yd".

Group: LCD Graphic 22

3.349 gMoveToRel

Procedure *gMoveToRel (Xr, Yr : integer);*
Bewegt den virtuellen Zeichen Cursor relativ zu seiner alten Position.

Group: LCD Graphic 22

3.350 gOpenView

Procedure *gOpenView (ViewPort : byte; Xs, Ys, Xe, Ye : integer) : boolean;*
Bestimmt die Position und Grösse eines ViewPorts in physischen Pixeln.

Group: LCD Graphic 22

3.351 gPntToScale

Procedure *gPntToScale (ViewPort: byte; Xp, Yp : integer; **var** XL, YL : integer);*
Wandelt eine Display absolute Koordinate in eine ViewPort relative Koordinate.

Group: LCD Graphic 22

3.352 gRestoreView

Procedure *gRestoreView (source : pointer);*
Uberschreibt das aktuelle ViewPort mit dem Inhalt des Speichers.

Group: LCD Graphic 22

3.353 gSaveView

Procedure *gSaveView (dest : pointer);*
Speichert den Inhalt des aktuellen ViewPorts in das RAM.

Group: LCD Graphic 22

3.354 gScaleToPnt

Procedure *gScaleToPnt (ViewPort: byte; XL, YL : integer;* ***var*** *Xp, Yp : integer);*
Wandelt eine ViewPort relative Koordinate in eine Display absolute Koordinate.

Group: LCD Graphic 22

3.355 gScaleView

Function *gScaleView (ViewPort : byte; Xs, Ys, Xe, Ye : integer) : boolean;*
Bestimmt die interne Skalierung des ViewPorts.

Group: LCD Graphic 22

3.356 gSetBitMapRAM

Procedure *gSetBitMapRAM (RAM: boolean);*
Schaltet zur Laufzeit zwischen BitMaps im ROM und im RAM um.

Group: LCD Graphic 22

3.357 gSetCharSet

Procedure *gSetCharSet (source : pointer);*
Bestimmt den aktuellen Zeichensatz (5x7).

Group: LCD Graphic 22

3.358 gSetCharSetRAM

Procedure *gSetCharSetRAM (RAM : boolean);*
Schaltet zur Laufzeit zwischen RAM und ROM Zeichensätzen um.

Group: LCD Graphic 22

3.359 gSetLineColor

Procedure *gSetLineColor (c : byte);*
Bestimmt die Farbe für das aktuelle ViewPort.

Für den Treiber muss Farbe importiert werden:
From *LCDGraphic* ***Import*** *GraphColor;*

Group: LCD Graphic 22

3.360 gSetLineMode

Procedure *gSetLineMode (LineWriteMode : TWriteMode);*
Bestimmt den Linien Schreib-Modus für das aktuelle ViewPort.

Group: LCD Graphic 22

3.361 gSetPixel

Procedure *gSetPixel (Px, Py : integer);*
Setzt einen Pixel an der Stelle „Px,Py" im aktuellen ViewPort.

Group: LCD Graphic 22

3.362 gSetTextBkGnd

Procedure *gSetTextBkGnd (backgnd : TTextBkGnd);*
Bestimmt den Text Hintergrund für das aktuelle ViewPort.

Group: LCD Graphic 22

3.363 gSetTextColor

Procedure *gSetTextColor (c : byte);*
Bestimmt die Farbe für das aktuelle ViewPort.

Für den Treiber muss Farbe importiert werden:
From *LCDGraphic* ***Import*** *GraphColor;*

Group: LCD Graphic 22

3.364 gSetTextJustify

Procedure *gSetTextJustify (Horiz : TtxtAlHor; Vert : TTxtAlVert);*
Bestimmt die Text Ausrichtung (Alignment) für das aktuelle ViewPort.

Group: LCD Graphic 22

3.365 gSetTextMode

Procedure *gSetTextMode (TextWriteMode : TWriteMode);*
Bestimmt den Text Schreibmodus für das aktuelle ViewPort.

Group: LCD Graphic 22

3.366 gSwitchView

Procedure *gSwitchView (ViewPort: byte) : boolean;*
Bestimmt das aktuelle ViewPort.

Group: LCD Graphic 22

3.367 gXorPixel

***rocedure** gXorPixel (Px, Py : integer);*
Invertiert einen Pixel an der Stelle „Px,Py" im aktuellen ViewPort.

Group: LCD Graphic 22

3.368 HexToInt

***Function** HexToInt (st : string) : integer [int8, byte, word, longint, longword];*
Konvertiert einen Hex-String, der aber *kein* "$" enthalten darf.

Group: Strings 6

3.369 Hi

***Function** Hi (w : word|integer) : byte;*
Gibt das höherwertige Byte eines 16bit Wertes zurück.

Group: Maths 5

3.370 Higher

***Function** Higher (x, y : type) : type;*
Gibt den grösseren zweier Werte zurück.Die Typen der beiden Argumente müssen identisch sein.
type: Byte, Int8, Word, Integer, Longint, Longword, Int64, Word64, Fix64, Float.

Group: Maths 5

3.371 HiNibble

***Function** HiNibble (w : byte|int8) : byte|int8;*
Gibt die höherwertigen 4 Bit eines Bytes zurück.

Group: Maths 5

3.372 HiWord

***Function** HiWord (ww : Longword|longint) : word|integer;*
Gibt das höherwertige Word eines 32bit Wertes zurück.

Group: Maths 5

3.373 I2C_Disp7Clear

***Procedure** I2C_Disp7Clear (**const** Disp : TI2C_Disp7);*
Das ganze Display wird gelöscht.

Group: I2C Disp7 12

3.374 I2C_Disp7ClEOL

Procedure *I2C_Disp7ClEOL (**const** Disp : TI2C_Disp7);*
Das Display wird von der aktuellen Cursorposition bis Zeilenende gelöscht.

Group: I2C Disp7

3.375 I2C_Disp7Ctrl

Function *I2C_Disp7Ctrl (**const** Disp : TI2C_Disp7; **const** ctrl : TI2C_Ctrl7);*
Schaltet alle Segmente ein/aus.

Group: I2C Disp7

3.376 I2C_Disp7DigitBlink

Procedure *I2C_Disp7DigitBlink (Disp : TI2C_Disp7; digit: byte; blink: boolean);*
Lässt ein Digit blinken.

Group: I2C Disp7

3.377 I2C_Disp7Dim

Procedure *I2C_Disp7Dim (**const** Disp : TI2C_Disp7; **const** dim : byte);*
Das Display wird gedimmt.

Group: I2C Disp7

3.378 I2C_Disp7Get

Function *I2C_Disp7Get : TI2C_Disp7;*
Liefert als Ergebnis das aktuell eingestellt Display.

Group: I2C Disp7

3.379 I2C_Disp7Init

Function *I2C_Disp7Init (**const** Disp : TI2C_Disp7; BlinkRate, DutyCycle : byte);*
Initialisiert das Display.

Group: I2C Disp7

3.380 I2C_Disp7Out

Procedure *I2C_Disp7Out (**const** ch : char);*
Schreibt in den Display Buffer.

Group: I2C Disp7

3.381 I2C_Disp7Pos

Procedure *I2C_Disp7Pos (**const** Disp : TI2C_Disp7; **const** digit : byte);*
Der Schreibcursor wird an eine bestimmte Stelle positioniert.

Group: I2C Disp7 12

3.382 I2C_Disp7Refresh

Procedure *I2C_Disp7Refresh (**const** Disp : TI2C_Disp7);*
Erzwingt ein Update des Displays.

Group: I2C Disp7 12

3.383 I2C_Disp7Set

Procedure *I2C_Disp7Set (**const** disp : TI2C_Disp7);*
Optionales Umschalten des Treibers auf ein bestimmtes Display.

Group: I2C Disp7 12

3.384 I2CexpStat

Function *I2CexpStat (Port: TI2Cport) : boolean;*
Prüft den Status eines I2C-PortSlaves.

Group: I2C Expand 13

3.385 I2CexpStat_5

Function *I2CexpStat_5 (Port: TI2Cport) : boolean;*
Prüft den Status eines I2C-PortSlaves.

Group: I2Cexpand_5 14

3.386 I2Cinp

Function *I2Cinp (SlaveAdr : byte; **var** Data) : boolean;*
Liest mindestens ein Byte aus dem angewählten Slave.

Group: I2C Port 13

3.387 I2Cout

Function *I2Cout (const SlaveAdr : byte, Command : byte|word [; Data]) : boolean;*
Schreibt mindestens ein Byte in den angewählten Slave.

Group: I2C Port 13

3.388 I2Cstat

Function *I2Cstat (SlaveAdr : byte) : boolean;*
Prüft den Status eines I2C-Slaves.

Group: I2C Port 13

3.389 Inc

Procedure *Inc (var v [, step] : type);*
Variable inkrementieren.

Group: Maths 5

3.390 Incl

Procedure *Incl (v : byte; num: byte);*
Setzt ein Bit in einem Byte.

Procedure *Incl (SrcDest : BitSet; op : BitSet);*
BitSet in einem BitSet setzen.

Group: System 7

3.391 IncrCount4start

Procedure *IncrCount4start;*
Startet den Scan Timer.

Group: Increment Counter 4chan 15

3.392 IncrCount4stop

Procedure *IncrCount4stop;*
Stoppt den Scan Timer.

Group: Increment Counter 4chan 15

3.393 IncSema

Procedure *IncSema (s : semaphore);*
Die Semaphore wird um eins erhöht.

Group: System 7

3.394 IncToLim

Function *IncToLim (**var** v : ordinal [, limit : ordinal[; val : ordinal]]) : boolean;*
"v" wird erhöht, vorrausgesetzt "v" hat noch nicht seine natürliche Grenze erreicht.

Group: Maths 5

3.395 IncToLimWrap

Function *IncToLimWrap (**var** value, lim, pres : type) : boolean;*
Inkrementiert die Variable "value" jeweils um 1.

Group: Maths 5

3.396 Inp_Raise

Function *Inp_Raise1 (bit : byte) : boolean;*
Function *Inp_Raise2 (bit : byte) : boolean;*
Function *Inp_RaiseG (bit : byte) : boolean;*
Gibt ein True zurück, wenn der Pin "bit" aktiviert wurde.

Group: SwitchPorts 39

3.397 Inp_Stable

Function *Inp_Stable1 (bit : byte) : boolean;*
Function *Inp_Stable2 (bit : byte) : boolean;*
Function *Inp_StableG (bit : byte) : boolean;*
Gibt ein True zurück, wenn der Pin "bit" aktiv ist.

Group: SwitchPorts 39

3.398 Insert

Procedure *Insert (src : string; **var** dst : string; p : byte);*
Fügt ein String in einen String ein.

Group: Strings 6

3.399 Int

Function *Int (f : float|fix64) : float|fix64;*
liefert den Integerteil des Arguments zurück.

Group: Maths 5

3.400 IntegrateB

Function *IntegrateB (oldVal, newVal, fact : byte) : byte;*
Integriert einen neuen Wert zu einem schon vorhandenen.

Group: Maths 5

3.401 IntegrateI

Function *IntegrateI (oldVal, newVal : integer; fact : byte) : integer;*
Integriert einen neuen Wert zu einem schon vorhandenen.

Group: Maths 5

3.402 IntegrateI8

Function *IntegrateI8 (oldVal, newVal : int8; fact : byte) : int8;*
Integriert einen neuen Wert zu einem schon vorhandenen.

Group: Maths 5

3.403 IntegrateW

Function *IntegrateW (oldVal, newVal : word; fact : byte) : word;*
Integriert einen neuen Wert zu einem schon vorhandenen.

Group: Maths 5

3.404 InterPolX

Function *InterPolX (**const** LookUp : pointer; x: type; **var** y: type) : boolean;*
Interpoliert einen Wert mit einer Tabelle.

Group: Diverse 4

3.405 InterPolY

Function *InterPolY (**const** LookUp : pointer; y: type; **var** x: type) : boolean;*
Interpoliert einen Wert mit einer Tabelle.

Group: Diverse 4

3.406 IntToBin

Function *IntToBin (value : word|integer) : string;*
Gibt die Repräsentation der Bits des Arguments in einem String.

Group: Strings 6

3.407 IntToFix64

Function *IntToFix64(i : ordinal) : fix64;*
wandelt einen ordinal Wert (Byte…LongInt) in ein Fix64 um.

Groups: Maths 5, Fix64 4

3.408 IntToHex

Function *IntToHex (i : integer) : string;*
Konvertiert 16bit Wert in einen hex-String.

Group: Strings 6

3.409 IntToStr

Function *IntToStr (i : word) : string;*
Konvertiert Numerischen 16bit Wert in einen String.

Group: Strings 6

3.410 IOexpUpdate

Procedure *IOexpUpdate;*
Veranlasst den Treiber alle Input Schieberegister einzulesen
und in dem Input Array abzulegen.

Group: IOexpand 16

3.411 IPtoStr

Function *IPtoStr (IP : TIPAddr) : String[15];*
Konvertiert ein IP-Adress Array in einen String "aaa:bbb:ccc:ddd".

Group: Strings 6 TINA TCP/IP 41 WizNet TCP/IP 45

3.412 IsCurProcess

Function *isCurProcess (ID : byte|Name : ProcName) : boolean;*
Abfrage ob der aktuelle Prozess/Task die ID oder den Namen hat.

Group: MultiTasking 5

3.413 IsPowOfTwo

Function *IsPowOfTwo (n : type) : boolean;*
Prüft die Zahl "n" ob sie eine Zweier Potenz repräsentiert.

Group: Maths 5

3.414 isSysTimerZero

Function *isSysTimerZero (tm : tSysTimer) : boolean;*
Gibt true zurück wenn der Timer den Wert 0 hat.

Group: Diverse 4

3.415 KeyBoardEnable

Procedure *KeyBoardEnable (ena : boolean);*
Sperrt oder gibt das komplette KeyBoard frei.

Group: KeyBoard 4x4 16

3.416 KeyBoardEnable8

Procedure *KeyBoardEnable8 (ena : boolean);*
Sperrt oder gibt das komplette KeyBoard frei.

Group: KeyBoard 8x8 17

3.417 KeyBoardRepeat

Procedure *KeyBoardRepeat (rept : boolean);*
Sperrt oder gibt die Repeat Funktion frei.

Group: KeyBoard 4x4 16

3.418 KeyBoardRepeat8

Procedure *KeyBoardRepeat8 (rept : boolean);*
Sperrt oder gibt die Repeat Funktion frei.

Group: KeyBoard 8x8 17

3.419 KeyRaised

Function *KeyRaised (**const** key : Keys) : boolean;*
Ergibt true wenn die Taste gedrückt wurde.

Group: KeyBoard 4x4 16

3.420 KeyRaised8

Function *KeyRaised8 (**const** key : Keys) : boolean;*
Ergibt true wenn die Taste gedrückt wurde.

Group: KeyBoard 8x8 17

3.421 KeyStat

Function *KeyStat : boolean;*
Der Status des KeyBoards wird zurückgegeben.

Group: KeyBoard 4x4 16

3.422 KeyStat8

Function *KeyStat8 : boolean;*
Der Status des KeyBoards wird zurückgegeben.

Group: KeyBoard 8x8 17

3.423 KeyStatRaised

Function *KeyStatRaised : boolean;*
Der Status des KeyBoards wird zurückgegeben.

Group: KeyBoard 4x4 16

3.424 KeyStatRaised8

Function *KeyStatRaised8 : boolean;*
Der Status des KeyBoards wird zurückgegeben.

Group: KeyBoard 8x8 17

3.425 LANrxAutoAck

Procedure *LANrxAutoAck (**const** OnOff : boolean);*
Steuert das automatische Acknowledge Verfahren.

Group: Serial LAN 29

3.426 LANrxClear

Procedure *LANrxClear;*
Das Statusbyte des RxBuffers wird zurückgesetzt.

Group: Serial LAN 29

3.427 LANrxStat

Function *LANrxStat : boolean;*
Gibt ein true zurück, wenn ein Frame empfangen wurde.

Group: Serial LAN 29

3.428 LANtxClear

Procedure *LANtxClear;*
Das Statusbyte des TxBuffers wird zurückgesetzt.

Group: Serial LAN 29

3.429 LANtxFrame

Function *LANtxFrame (node : byte[word]; len : byte[word]) : boolean;*
Sendet einen Buffer Frame.

Group: Serial LAN 29

3.430 LANtxStat

Function *LANtxStat : boolean;*
Prüft, ob ein zu sendender Frame gesendet ist oder nicht.

Group: Serial LAN 29

3.431 LCDbarInit_M

Procedure *LCDbarInit_M;*
Initalisiert den BarGraph Treiber.

Group: LCD Bargraph 17 LCD Display Multi 18

3.432 LCDbarInit_P

Procedure *LCDbarInit_P;*
Initalisiert den BarGraph Treiber.

Group: LCD Bargraph 17 LCD Display 18

3.433 LCDbarOut

Procedure *LCDbarOut1 (**const** b : byte);*
Procedure *LCDbarOut2 (**const** b : byte);*
Procedure *LCDbarOut3 (**const** b : byte);*
Procedure *LCDbarOut4 (**const** b : byte);*
Setzt den BarGraphen auf einen neuen Wert und zeigt ihn an.

Groups: LCD Bargraph 17 LCD Display 18 LCD Display Multi 18

3.434 LCDbarSet

Procedure *LCDbarSet1 (**const** Line, PosA, Len, Scal : byte);*
Procedure *LCDbarSet2 (**const** Line, PosA, Len, Scal : byte);*
Procedure *LCDbarSet3 (**const** Line, PosA, Len, Scal : byte);*
Procedure *LCDbarSet4 (**const** Line, PosA, Len, Scal : byte);*
Initialisiert das Koordinaten System des BarGraphen.

Groups: LCD Bargraph 17 LCD Display 18 LCD Display Multi 18

3.435 LCDcharset

Procedure *LCDcharset (loc : char; b1, b2, b3, b4, b5, b6, b7, b8 : byte);*
Lädt ein selbst definiertes Grafik-Zeichen in den Character Generator.

Group: LCD Display 18

3.436 LCDcharset_M

Procedure *LCDcharset_M (LCD_num : TLCD_num; loc : char; c1, c2, c3, c4, c5, c6, c7, c8 : byte);*
Lädt ein selbst definiertes Grafik-Zeichen in den Character Generator.

Group: LCD Display Multi 18

3.437 LCDcharset_MP

Procedure *LCDcharset_MP (LCD_num : TLCD_num; loc : char; srcArea : byte; ptr : pointer);*
Lädt ein selbst definiertes Grafik-Zeichen in den Character Generator.

Group: LCD Display Multi 18

3.438 LCDcharsetP

Procedure *LCDcharsetP (**const** loc : char; srcArea : byte; ptr : pointer);*
Lädt ein selbst definiertes Grafik-Zeichen in den Character Generator.

Group: LCD Display 18

3.439 LCDclr

Procedure *LCDclr;*
Löscht das ganze Display und setzt den Cursor auf Position 0,0.

Group: LCD Display 18

3.440 LCDclr_M

Procedure *LCDclr_M (**const** LCD_num : TLCD_num);*
Löscht das ganze Display und setzt den Cursor auf Position 0,0.

Group: LCD Display Multi 18

3.441 LCDclrEOL

Procedure *LCDclrEOL;*
Löscht die aktuelle Zeile bis Zeilen Ende.

Group: LCD Display 18

3.442 LCDclrEOL_M

***Procedure** LCDclrEOL_M (**const** LCD_num : TLCD_num);*
Löscht die aktuelle Zeile ab Cursor Position bis Zeilen Ende.

Group: LCD Display Multi 18

3.443 LCDclrLine

***Procedure** LCDclrLine (Line : byte);*
Löscht die angegebene Zeile und setzt den Cursor auf Zeilen Anfang.

Group: LCD Display 18

3.444 LCDclrLine_M

***Procedure** LCDclrLine_M (**const** LCD_num : TLCD_num; **const** line : byte);*
Löscht die angegebene Zeile(0..n) und setzt den Cursor auf Zeilen Anfang.

Group: LCD Display Multi 18

3.445 LCDctrl

***Procedure** LCDctrl (**const** b : byte);*
Schreiben auf LCD-ControlPort mit *RS=0* und *RW=0*.

Group: LCD Display 18

3.446 LCDctrl_M

***Procedure** LCDctrl_M (**const** LCD_num : TLCD_num; **const** b : byte);*
Schreiben auf LCD-ControlPort mit *RS=0* und *RW=0*.

Group: LCD Display Multi 18

3.447 LCDcursor

***Procedure** LCDcursor (on, blink : boolean);*
Einstellung des Cursors Modus.

Group: LCD Display 18

3.448 LCDcursor_M

***Procedure** LCDcursor_M (**const** LCD_num : TLCD_num; on, blink : boolean);*
Einschalten des Displays und Einstellung des Cursors Modus.

Group: LCD Display Multi 18

3.449 LCDgetPort_M

Function *LCDgetPort_M : TLCD_num;*
Liefert als Ergebnis das aktuell eingestellt LCD-Port.

Group: LCD Display Multi 18

3.450 LCDgetXY

Function *LCDgetXY : word;*
Liest die aktuelle Cursor Position.

Group: LCD Display 18

3.451 LCDgetXY_M

Function *LCDgetXY_M (**const** LCD_num : TLCD_num) : word;*
Position des Cursors auslesen.

Group: LCD Display Multi 18

3.452 LCDhome

Procedure *LCDhome;*
Setzt den Cursor auf Position 0,0.

Group: LCD Display 18

3.453 LCDhome_M

Procedure *LCDhome_M (**const** LCD_num : TLCD_num);*
Setzt den Cursor auf Position 0,0.

Group: LCD Display Multi 18

3.454 LCDinp

Function *LCDinp : byte;*
Lesen des LCD DD-Ram bzw. CG-Ram mit *RS=1* und *RW=1.*

Group: LCD Display 18

3.455 LCDinp_M

Function *LCDinp_M (**const** LCD_num : TLCD_num) : byte;*
Lesen des LCD DD-Ram bzw. CG-Ram mit *RS=1* und *RW=1*.

Group: LCD Display Multi 18

3.456 LCDlower

Procedure *LCDlower;*
Umschalten zwischen zwei Display Controllern.

Group: LCD Display 18

3.457 LCDoff

Procedure *LCDoff;*
Schaltet das Display ab, verändert ansonsten nichts.

Group: LCD Display 18

3.458 LCDoff_M

Procedure *LCDoff_M (**const** LCD_num : TLCD_num);*
Schaltet das Display ab, verändert ansonsten nichts.

Group: LCD Display Multi 18

3.459 LCDon

Procedure *LCDon;*
Schaltet das Displays ein setzt den Cursor auf ON und BLINK.

Group: LCD Display 18

3.460 LCDon_M

Procedure *LCDon_M (**const** LCD_num : TLCD_num);*
Schaltet das Displays ein setzt den Cursor auf ON und BLINK.

Group: LCD Display Multi 18

3.461 LCDout

Procedure *LCDout (b : byte|char);*
Schreiben ins LCD Display-Ram.

Group: LCD Display 18

3.462 LCDout_M

Procedure *LCDout_M (**const** c : char|byte);*
Schreiben ins LCD Display-Ram.

Group: LCD Display Multi 18

3.463 LCDportInp_M

Function *LCDportInp_M (**const** LCD_num : TLCD_num) : byte;*
Diese Funktion liest die oberen 5 Input Pins des Control Ports.

Group: LCD Display Multi 18

3.464 LCDsetPort_M

Procedure *LCDsetPort_M (**const** LCD_num : TLCD_num);*
Selektiert das gewünschte Display.

Group: LCD Display Multi 18

3.465 LCDsetup

Procedure *LCDsetup;*
Initialisiert das Display.

Group: LCD Display 18

3.466 LCDsetup_M

Function *LCDsetup_M (**const** LCD_num : TLCD_num) : boolean;*
Initialisiert das Display.

Group: LCD Display Multi 18

3.467 LCDstat

Function *LCDstat : byte;*
Lesen des LCD StatusPorts mit *RS=0* und *RW=1*.

Group: LCD Display 18

3.468 LCDstat_M

Function *LCDstat_M (**const** LCD_num : TLCD_num) : byte;*
Lesen des LCD Status Ports mit *RS=0* und *RW=1*.

Group: LCD Display Multi 18

3.469 LCDupper

Procedure *LCDupper;*
Umschalten zwischen zwei Display Controllern.

Group: LCD Display 18

3.470 LCDxy

Procedure *LCDxy (column, row : byte);*
Positionierung des Cursors auf Spalte[x] und Zeile[y].

Group: LCD Display 18

3.471 LCDxy_M

Procedure *LCDxy_M (**const** LCD_num : TLCD_num; x, y : byte);*
Positionierung des Cursors auf Spalte[x] und Zeile[y].

Group: LCD Display Multi 18

3.472 LEDdotBlink

LEDdotBlink
T.B.D.

Group: LED DOT Display 23

3.473 LEDdotBlinkDigit

LEDdotBlinkDigit
T.B.D.

Group: LED DOT Display 23

3.474 LEDdotCharset

LEDdotCharset
T.B.D.

Group: LED DOT Display 23

3.475 LEDdotCharsetP

LEDdotCharsetP
T.B.D.

Group: LED DOT Display 23

3.476 LEDdotClr

LEDdotClr
T.B.D.

Group: LED DOT Display 23

3.477 LEDdotClrEOL

LEDdotClrEOL
T.B.D.

Group: LED DOT Display

3.478 LEDdotClrLine

LEDdotClrLine
T.B.D.

Group: LED DOT Display

3.479 LEDdotCursor

LEDdotCursor
T.B.D.

Group: LED DOT Display

3.480 LEDdotDim

LEDdotDim
T.B.D.

Group: LED DOT Display

3.481 LEDdotGetXY

LEDdotGetXY
T.B.D.

Group: LED DOT Display

3.482 LEDdotHome

LEDdotHome
T.B.D.

Group: LED DOT Display

3.483 LEDdotInit

LEDdotInit
T.B.D.

Group: LED DOT Display

3.484 LEDdotOff

LEDdotOff
T.B.D.

Group: LED DOT Display 23

3.485 LEDdotOn

LEDdotOn
T.B.D.

Group: LED DOT Display 23

3.486 LEDdotOut

LEDdotOut
T.B.D.

Group: LED DOT Display 23

3.487 LEDdotXY

LEDdotXY
T.B.D.

Group: LED DOT Display 23

3.488 Length

Function *Length (s : string) : byte;*
Gibt die aktuell belegte Länge eines Strings zurück.

Group: Strings 6

3.489 Lo

Function *Lo (w : word) : byte;*
Ergibt das niederwertige Byte eines 16bit Wertes.

Group: Maths 5

3.490 Lock

Procedure *Lock (p : process);*
Die ganze Rechenzeit des Prozessors wird einem Prozess zur Verfügung gestellt.

Group: MultiTasking 5

3.491 Log10

Function *Log10 (f : float) : float;*
Liefert als Ergebnis den dekadischen Logarithmus zurück.

Group: Maths 5

3.492 LogN

Function *LogN(f : float) : float;*
Liefert als Ergebnis den Logarithmus Naturalis zurück. Euler'sche Zahl e = 2.71...

Group: Maths 5

3.493 Long64ToHex

Function *Long64ToHex (**const** ii : Int64|Word64) : string;*
Konvertiert einen numerischen 64bit Wert in einen hex-String.

Group: Strings 6

3.494 Long64ToStr

Function *Long64ToStr (**const** ii : Int64|Word64[: **const** len : byte[: **const** space : char]]) : string;*
Konvertiert einen numerischen 64bit Wert in einen String.

Group: Strings 6

3.495 LongAsFloat

Function *LongAsFloat (L : LongWord) : float;*
Wandelt das Argument in einen Float.
Führt jedoch **keine Typ Konvertierung** durch!

Group: Diverse 4

3.496 LongToHex

Function *LongToHex (w : longword) : string;*
Konvertiert einen numerischen 32bit Wert in einen hex-String.

Group: Strings 6

3.497 LongToStr

Function *LongToStr (ii : longword) : string;*
Konvertiert einen numerischen 32bit Wert in einen String.

Group: Strings 6

3.498 LoNibble

Function *LoNibble (w : byte|int8) : byte|int8;*
Gibt die niederwertigen 4 Bit eines Bytes zurück.

Group: Maths 5

3.499 LowCase

Function *LowCase (ch : char) : char;*
Wandelt einen Buchstaben in einen Kleinbuchstaben.

Group: Strings 6

3.500 Lower

Function *Lower (x, y : type) : type;*
Gibt den kleineren zweier Werte zurück. Die Typen der beiden Argumente müssen identisch sein.
type: Byte, Int8, Word, Integer, Longint, Longword, Int64, Word64, Fix64, Float.

Group: Maths 5

3.501 LowerCase

Function *LowerCase (st : string) : string;*
Wandelt String in Kleinbuchstaben.

Group: Strings 6

3.502 LoWord

Function *LoWord (v : Longword|longint) : word [integer];*
Liefert das niederwertiges Word eines 32bit Wertes (LongInt/LongWord).

Group: Maths 5

3.503 LowPassFW

LowPassFW
T.B.D.

Group: Maths 5

3.504 LPTctrl

Procedure *LPTctrl (ctrl : tLPTlineSet);*
Steuert das Controlport bzw. die Steuerleitungen für den Printer.

Group: LPT Printer 23

3.505 LPTdir

Procedure *LPTdir (inp : boolean);*
Steuert die Datenrichtung des Datenports. True = output, false = input.

Group: LPT Printer 23

3.506 LPTinit

Procedure *LPTinit;*
Initialisiert den Printer durch die Init und die Select Leitung.

Group: LPT Printer 23

3.507 LPTinp

Function *LPTinp : byte;*
Liest das Datenport.

Group: LPT Printer 23

3.508 LPTout

Procedure *LPTout (dat : byte);*
Dies ist die Druck Funktion.

Group: LPT Printer 23

3.509 LPTreset

Procedure *LPTreset;*
Gibt einen kurzen Impuls auf die Init-Leitung.

Group: LPT Printer 23

3.510 LPTstat

Function *LPTstat : tLPTlineset;*
Gibt den Status des Control Ports zurück.

Group: LPT Printer 23

3.511 MACtoStr

Function *MACtoStr (MAC : TMACAddr) : String[17];*
Konvertiert ein MAC-Adress Array zu einem String der Form "aa:bb:cc:dd:ee:ff".

Group: Strings 6 TINA TCP/IP 41 WizNet TCP/IP 45

3.512 Max

Function *Max (x: type) : type;*
Ergibt den grössten möglichen Wert des Typs.

Group: Maths 5

3.513 mb_GetModBusDevID

Function *mb_GetModBusDevID: byte;*
Gibt die Kennung (ID) des Treibers/Geräts zurück.

Group: ModBus ASCII 25 ModBus RTU 26

3.514 mb_GetModBusExceptionStatus

Function *mb_GetModBusExceptionStatus: byte;*
Liest den ExceptionStatus.

Group: ModBus RTU 26

3.515 mb_GetModBusTimeout

Function *mb_GetModBusTimeout: word;*
Liefert das eingestellte Time-Out zurüick (msec).

Group: ModBus ASCII 25 ModBus RTU 26

3.516 mb_SetAfterCoilRead

procedure *mb_SetAfterCoilRead (proc: tAfterCoilRead);*
Callback Funktion. Coils bearbeiten nachdem sie vom Client gelesen wurden.

Group: ModBus ASCII 25 ModBus RTU 26

3.517 mb_SetAfterCoilWrite

procedure *mb_SetAfterCoilWrite (proc: tAfterCoilWrite);*
Callback Funktion. Coils bearbeiten nachdem sie vom Client empfangen wurden.

Group: ModBus ASCII 25 ModBus RTU 26

3.518 mb_SetAfterRegisterRead

procedure *mb_SetAfterRegisterRead (proc: tAfterRegisterRead);*
Callback Funktion. Register bearbeiten nachdem sie zum Client gesendet wurden.

Group: ModBus ASCII 25 ModBus RTU 26

3.519 mb_SetAfterRegisterWrite

procedure *mb_SetAfterRegisterWrite (proc: tAfterRegisterWrite);*
Callback Funktion. Register bearbeiten nachdem sie vom Client empfangen wurden.

Group: ModBus ASCII 25 ModBus RTU 26

3.520 mb_SetBeforeCoilRead

procedure *mb_SetBeforeCoilRead (proc: tBeforeCoilRead);*
Callback Funktion. Coils bearbeiten bevor sie zum Client gesendet werden.

Group: ModBus ASCII 25 ModBus RTU 26

3.521 mb_SetBeforeCoilWrite

procedure *mb_SetBeforeCoilWrite (proc: tBeforeCoilWrite);*
Callback Funktion. Coils bearbeiten bevor sie vom Client empfangen werden.

Group: ModBus ASCII 25 ModBus RTU 26

3.522 mb_SetBeforeRegisterRead

procedure *mb_SetBeforeRegisterRead (proc: tBeforeRegisterRead);*
Callback Funktion. Register bearbeiten bevor sie zum Client gesendet werden.

Group: ModBus ASCII 25 ModBus RTU 26

3.523 mb_SetBeforeRegisterWrite

procedure *mb_SetBeforeRegisterWrite (proc: tBeforeRegisterWrite);*
Callback Funktion. Register bearbeiten bevor sie vom Client empfangen werden.

Group: ModBus ASCII 25 ModBus RTU 26

3.524 mb_SetModBusDevID

Procedure *mb_SetModBusDevID(id: byte);*
Jedes ModBus Gerät muss eine eindeutige Kennung (ID) haben.

Group: ModBus ASCII 25 ModBus RTU 26

3.525 mb_SetModBusExceptionStatus

Procedure *mb_SetModBusExceptionStatus (status: byte);*
Setzt den ExceptionStatus.

Group: ModBus RTU 26

3.526 mb_SetModBusTimeout

Procedure *mb_SetModBusTimeout (time: word); {set timeout in ms}*
Bestimmt das Time-Out zwischen zwei Rx-Bytes (msec).

Group: ModBus ASCII 25 ModBus RTU 26

3.527 mDelay

Procedure *mDelay (d : word);*
Software Delay in msec.

Group: System 7

3.528 Min

Function *Min (x: type) : type;*
Ergibt den kleinsten möglichen Wert des Typs.

Group: Maths 5

3.529 Mirror16

Function *Mirror16 (w : word|integer) : word|integer;*
Spiegelt das Argument. Tauscht Bit15 <-> Bit0, Bit14 <-> Bit1, ...

Group: Maths 5

3.530 Mirror32

Function *Mirror32 (Lw : longword|longint) : longword|longint;*
Spiegelt das Argument. Tauscht Bit31 <-> Bit0, Bit30 <-> Bit1, ...

Group: Maths 5

3.531 Mirror8

Function *Mirror8 (b : int8|byte|char) : int8|byte|char;*
Spiegelt das Argument. Tauscht Bit7 <-> Bit0, Bit6 <-> Bit1, ...

Group: Maths 5

3.532 MRFgetLostPkts

Function *mrfGetLostPkts : byte;*
gibt die Anzahl der trotz Retries nicht zustellbaren Pakete zurück.
Gleichzeitig wird der interne Counter im 24L01 zurückgesetzt.

Group: MIRF24 Port 24

3.533 MRFgetRetryCnt

Function *mrfGetRetryCnt : byte;*
Nach einer erfolgreichen Sende Operation kann damit die Anzahl der benötigten Retries ausgelesen werden. Je höher dieser Wert ist, desto schlechter war die Verbindung.

Group: MIRF24 Port

3.534 MRFgetRxPower

Function *mrfGetRxPower : byte;*
liefert im Bit0 die Empfangsqualität zurück. 0 = schlechter Empfang, 1 = guter Empfang.

Group: MIRF24 Port

3.535 MRFgetRxType

Function *mrfGetRxType : tMRFpkt;*
dient zum Pollen des Empfangs. Gibt den Empfangs Status zurück.
tMRFpkt = (mrfPKTnone, mrfPKTdata, mrfPKTbcast).

Group: MIRF24 Port

3.536 MRFgetState

Function *mrfGetState : tMRFstat;*
gibt das Status Registers des 24L01 zurück.

tMRFstat = BitSet of enMRFstat.
enMRFstat = (mrfTX_full, mrfRX_pn0, mrfRX_pn1, mrfRX_pn2, mrfMAX_RT, mrfTX_DS, mrfRX_DR);

Wird normalerweise nicht gebraucht. Die Bedeutung der Bits ist dem 24L01 Datenblatt zu entnehmen.

Group: MIRF24 Port

3.537 MRFinit

Function *mrfInit : boolean;*
initialisiert den 24L01 mit den oben vorgegebenen Werten und stellt den Empfang aktiv. Bei einem Fehler wird ein false zurückgegeben.

Group: MIRF24 Port

3.538 MRFrxPacket

Function *mrfRxPacket (destPtr : pointer; TimeOut : byte; var recvd : byte) : tMRFpkt*
versucht ein Packet abzuholen.
destPtr muss auf eine RAM Datenstruktur mit der Grösse 32bytes zeigen.
TimeOut gibt die Zeit in msec an, die bis zum Erfolg bzw. Abbruch gewartet werden soll.
In recvd steht die tatsächlich empfangene Anzahl der Bytes (max. 32).
Das Ergebnis ist tMRFpkt = (mrfPKTnone, mrfPKTdata, mrfPKTbcast) .

Bei einem Timeout erfolgt ein mrfPKTnone. Wurde ein Standard Datenpaket empfangen, so kommt ein mrfPKTdata zurück und bei einem Broadcast ein mrfPKTbcast.

Group: MIRF24 Port

3.539 MRFsetChan

Procedure *mrfSetChan (chan : tMRFchan; wr : boolean);*
stellt den gewünschten RF Kanal im ISM Band ein.
chan wird mit der Enumeration tMRFchan angegeben (mrfChan1.. mrfChan14).
wr bestimmt dabei, ob der Kanal sofort eingestellt werden soll.
Vor dem ersten Init muss wr false sein.
Nach einem Init kann der Kanal zur Laufzeit mit wr = true umgestellt werden, ohne ein Init durchzuführen.
Alternativ kann auch die Prozedur MRFsetFreq benutzt werden.

Group: MIRF24 Port

3.540 MRFsetFreq

Procedure *mrfSetFreq (freq : word; wr : boolean);*
stellt die gewünschte Frequenz im ISM Band ein.
Frequenz freq wird in MHz angegeben. Gültige Werte sind dabei 2400.. 2484.
wr bestimmt dabei, ob die Frequenz sofort eingestellt werden soll.
Vor dem ersten Init muss wr false sein.
Nach einem Init kann die Frequenz zur Laufzeit mit wr = true umgestellt werden, ohne ein Init durchzuführen.
Alternativ kann auch die Prozedur MRFsetChan benutzt werden.

Group: MIRF24 Port

3.541 MRFsetLocalAdr

Procedure *mrfSetLocalAddr (adr : byte);*
stellt die lokale Adresse (0..255) ein.
Das ist die logische Adresse mit der andere Nodes diesen Node ansprechen müssen.
Diese Adresse muss unique sein, d.h. sie darf nur einmal im Netzwerk vorkommen.

Group: MIRF24 Port

3.542 MRFsetPower

procedure *mrfSetPower (pwr : tMRFpwr);*
stellt die gewünschte Ausgangsleistung (RF power) ein.
Die Power wird mit der Enumeration tMRFpwr angegeben (mrfdBm0.. mrfdBm18).
mrfdBm0 ist max. Power und mrfdBm18 ist minimal Power.

Group: MIRF24 Port

3.543 MRFsetPWRdown

Procedure *mrfSetPWRdown;*
Die Prozedur schaltet den Controller 24L01 in den Power-down Mode.
Für einen Neustart genügt dann der Aufruf der Funktion MRFinit

Group: MIRF24 Port

3.544 MRFsetRetryMax

Procedure *mrfSetRetryMax(rmax : byte);*
stellt die max. Retry Anzahl ein (0..15).
Tritt beim Datentransfer ein Fehler auf, z.B. das Acknowledge des Empfängers kommt nicht rechtzeitig oder gar nicht, dann wiederholt der Sender das Telegramm bis zu rmax mal. Fehlt das ACK dann immer noch,
kommt die Sendefunktion mrfTxPacket mit einem false zurück.

Group: MIRF24 Port

3.545 MRFsetRetryTimeOut

Procedure *mrfSetRetryTimeOut (tmo : byte);*
stellt das Delay zwischen zwei Retries ein (1..15). Diese Zeit rechnet sich so:
250usec + (tmo * 250usec). Typische Werte sollten bei 1msec liegen (tmo = 3..4).

Group: MIRF24 Port

3.546 MRFsetRFspeed

Procedure *mrfSetRFspeed(spd : tMRFrfSpeed);*
stellt die Datenrate "on air" ein. Die Rate wird mit der Enumeration tMRFrfSpeed angegeben (mrfRF250, mrfRF1000, mrfRF2000).

Group: MIRF24 Port

3.547 MRFtxPacket

Function *mrfTxPacket (adr : byte; srcPtr : pointer; cnt : byte; bc : boolean) : boolean;*
versucht ein Daten Paket oder ein Broadcast Paket zu senden.

adr bezeichnet die logische Adresse (0..255) des gewünschten Empfängers (Node).
srcPtr muss auf die Quelle im RAM zeigen.
cnt bestimmt die Anzahl der zu sendenden Bytes (max. 32).
bc bestimmt ob das Paket ein Broadcast oder ein Daten Paket ist.
Bei einem Broadcast wird der Parameter adr ignoriert.
Das Ergebnis wird false wenn ein Hardware Fehler vorliegt (Data oder Broadcast).
Ebenfalls ein false kommt zurück (Data) wenn der Empfänger auch nach x Sende Retries kein ACK geschickt hat bzw. das ACK verloren ging.

Der MIRF24port Treiber bietet zwei Arten von Daten Paket Transfers an:

1. Broadcast.

Dieses Paket hat eine bestimmte Kennung, so dass alle erreichbaren MIRF24 Nodes dieses Paket empfangen und auswerten können.
In diesem Fall ist das Hardware ACK abgeschaltet und auch der Empfänger (Applikation) sollte nicht darauf antworten.

2. Data.
Dieses Paket geht nur an eine bestimmte Adresse (Node) und wird auch nur von diesem empfangen.
Der Empfänger (Node) schickt ein automatisches Hardware ACK an den Sender zurück und quittiert damit den korrekten Empfang.
Bei einem CRC Fehler etc. unterbleibt natürlich das ACK und der Sender muss ein Retry durchführen. Ein ACK vom Empfänger bedeutet zu diesem Zeitpunkt keinesfalls, dass der Empfänger das Paket schon aus dem 24L01 ausgelesen hat.
Solange dies nicht geschehen ist schickt der Empfänger keine weiteren ACKs für einkommende Datenpakete so dass der Sender nach Ablauf der eingestellten Retries aus der Sende Funktion mit einem Fehler zurückkehrt.

Achtung:
Vor Senden eines Broadcasts und danach sollte etwas Pause sein, so dass alle Empfänger bereit sind.

Group: MIRF24 Port 24

3.548 MSPIinOut

Procedure *MSPIinOut0 (source, dest : pointer; count : word);*
Procedure *MSPIinOut1 (source, dest : pointer; count : word);*
. . .
Group: ATMega MSPI HardwareDriver 35

XMega:
Procedure *MSPIinOut_C0 (source, dest : pointer; count : word);*
Procedure *MSPIinOut_C1 (source, dest : pointer; count : word);*
Procedure *MSPIinOut_D0 (source, dest : pointer; count : word);*
. . .
Group: XMega MSPI HardwareDriver 36

Liest und schreibt einen Datenblock.

3.549 MSPIinOutByte

Function *MSPIinOutByte0 (b : byte) : byte;*
Function *MSPIinOutByte1 (b : byte) : byte;*
. . .
Group: ATMega MSPI HardwareDriver 35

XMega:
Function *MSPIinOutByte_C0 (b : byte) : byte;*
Function *MSPIinOutByte_C1 (b : byte) : byte;*
Function *MSPIinOutByte_D0 (b : byte) : byte;*
. . .
Group: XMega MSPI HardwareDriver 36

Liest und schreibt ein Byte in den MSPI-Slave.

3.550 MSPIinp

Procedure *MSPIinp0 (dest : pointer; count : word);*
Procedure *MSPIinp1 (dest : pointer; count : word);*
. . .
Group: ATMega MSPI HardwareDriver 35

XMega:
Procedure *MSPIinp_C0 (dest : pointer; count : word);*
Procedure *MSPIinp_C1 (dest : pointer; count : word);*
Procedure *MSPIinp_D0 (dest : pointer; count : word);*
. . .
Group: XMega MSPI HardwareDriver 36

Liest einen Datenblock aus dem MSPI-Slave.

3.551 MSPIinpByte

Function *MSPIinpByte0 : byte;*
Function *MSPIinpByte1 : byte;*
. . .
Group: ATMega MSPI HardwareDriver 35

XMega:
Function *MSPIinpByte_C0 : byte;*
Function *MSPIinpByte_C1 : byte;*
Function *MSPIinpByte_D0 : byte;*
. . .
Group: XMega MSPI HardwareDriver 36

Liest ein Byte aus dem MSPI-Slave.

3.552 MSPIinpLong

Function *MSPIinpLong0 : longword;*
Function *MSPIinpLong1 : longword;*
. . .
Group: ATMega MSPI HardwareDriver 35

XMega:
Function *MSPIinpLong_C0 : longword;*
Function *MSPIinpLong_C1 : longword;*
Function *MSPIinpLong_D0 : longword;*
. . .
Group: XMega MSPI HardwareDriver 36

Liest ein LongWord aus dem MSPI-Slave.

3.553 MSPIinpWord

Function *MSPIinpWord0 : word;*
Function *MSPIinpWord1 : word;*
. . .
Group: ATMega MSPI HardwareDriver 35

XMega:
Function *MSPIinpWord_C0 : word;*
Function *MSPIinpWord_C1 : word;*
Function *MSPIinpWord_D0 : word;*
. . .
Group: XMega MSPI HardwareDriver 36

Liest ein Word aus dem MSPI-Slave.

3.554 MSPIout

Procedure *MSPIout0 (source : pointer; count : word);*
Procedure *MSPIout1 (source : pointer; count : word);*
. . .
Group: ATMega MSPI HardwareDriver 35

XMega:
Procedure *MSPIout_C0 (source : pointer; count : word);*
Procedure *MSPIout_C1 (source : pointer; count : word);*
Procedure *MSPIout_D0 (source : pointer; count : word);*
. . .
Group: XMega MSPI HardwareDriver 36

Schreibt einen Datenblock in den MSPI-Slave.

3.555 MSPIoutByte

Procedure *MSPIoutByte0 (b : byte);*
Procedure *MSPIoutByte1 (b : byte);*
. . .
Group: ATMega MSPI HardwareDriver 35

XMega:
Procedure *MSPIoutByte_C0 (b : byte);*
Procedure *MSPIoutByte_C1 (b : byte);*
Procedure *MSPIoutByte_D0 (b : byte);*
. . .
Group: XMega MSPI HardwareDriver 36

Schreibt ein Byte in den MSPI-Slave.

3.556 MSPIoutLong

Procedure *MSPIoutLong0 (L : longword);*
Procedure *MSPIoutLong1 (L : longword);*
. . .
Group: ATMega MSPI HardwareDriver 35

XMega:
Procedure *MSPIoutLong_C0 (l : longword);*
Procedure *MSPIoutLong_C1 (l : longword);*
Procedure *MSPIoutLong_D0 (l : longword);*
. . .
Group: XMega MSPI HardwareDriver 36

Schreibt ein LongWord in den MSPI-Slave.

3.557 MSPIoutWord

Procedure *MSPIoutWord0 (w : word);*
Procedure *MSPIoutWord1 (w : word);*
. . .
Group: ATMega MSPI HardwareDriver 35

XMega:
Procedure *MSPIoutWord_C0 (w : word);*
Procedure *MSPIoutWord_C1 (w : word);*
Procedure *MSPIoutWord_D0 (w : word);*
. . .
Group: XMega MSPI HardwareDriver 36

Schreibt ein Word in den MSPI-Slave.

3.558 MulDivByte

Function *MulDivByte (a1, a2, d : byte) : byte;*
Die Funktion errechnet das 16bit Ergebnis der Multiplikation und teilt dies durch den 8bit Divisor. Damit wird ein Overflow Fehler ausgeschlossen.

Group: Maths 5

3.559 MulDivInt

Function *MulDivInt (a1, a2, d : integer|word) : integer|word;*
Die Funktion errechnet das 32bit Ergebnis der Multiplikation und teilt dies durch den 16bit Divisor. Damit wird ein Overflow Fehler ausgeschlossen.

Group: Maths 5

3.560 MulDivInt8

Function *MulDivInt8 (a1, a2, d : int8) : int8;*
Die Funktion errechnet das 16bit Ergebnis der Multiplikation und teilt dies durch den 8bit Divisor. Damit wird ein Overflow Fehler ausgeschlossen.

Group: Maths 5

3.561 MulDivLong

Function *MulDivLong (a1, a2, d : longint|longword) : longint|longword;*
Die Funktion errechnet das 64bit Ergebnis der Multiplikation und teilt dies durch den 32bit Divisor. Damit wird ein Overflow Fehler ausgeschlossen.

Group: Maths 5

3.562 Negate

Function *Negate (v : type) : type;*
Liefert den negativen Wert (Two's Complement) eines Byte, Int8..Longword, LongInt, Word54, Int64, Fix64 oder Float

a:= Negate (a);

Group: Maths 5

3.563 NoInts

Procedure *NoInts;*
Sperrung der Interrupts ohne Beeinflussung des entsprechenden System Flag.

Group: System 7

3.564 NOP

Procedure *NOP;*
Es erzeugt ein Assembler "NOP".

Group: System 7

3.565 Odd

Function *Odd (x : type) : boolean;*
Wert auf ungeradzahlig testen.

Group: Maths 5

3.566 OnADCread

Procedure *OnADCread;*
Callback. Wird aufgerufen wenn im SysTick das ADC Ergebnis gelesen wird.

Group: ADC 8

3.567 OnFAT16_SS

Procedure *OnFAT16_SS;*
Callback. Zur Steuerung des SPI SS-Pin durch die Applikation.

Group: System 7

3.568 OnIdleProcess

Procedure *OnIdleProcess;*
Callback. Wird bei jedem start des Idle Processes aufgerufen.

Group: System 7

3.569 OnSchedulerEntry

Procedure *OnSchedulerEntry;*
Callback. Wird bei jedem Eintritt in den Scheduler aufgerufen.

Group: System 7

3.570 OnSchedulerExit

Procedure *OnSchedulerExit;*
Callback. Wird bei jedem Verlassen des Schedulers aufgerufen.

Group: System 7

3.571 OnSerRxResumed

Procedure *OnSerRxResumed1;*
Procedure *OnSerRxResumed2;*
. . .

XMega:
Procedure *OnSerRxResumedC0;*
Procedure *OnSerRxResumedC1;*
Procedure *OnSerRxResumedD0;*
. . .

Callback. Wird aufgerufen der SerPort Treiber den RxBuffer durch senden eines XON oder de-aktivieren der DSR Handshake Leitung wieder freigibt

Group: SerPort 30

3.572 OnSerRxStopped

Procedure *OnSerRxStopped1;*
Procedure *OnSerRxStopped2;*
. . .

XMega:
Procedure *OnSerRxStoppedC0;*
Procedure *OnSerRxStoppedC1;*
Procedure *OnSerRxStoppedC0;*
. . .

Callback. Wird aufgerufen wenn der SerPort Treiber den RxBuffer durch senden eines XOFF oder aktivieren der DSR Handshake Leitung sperrt.

Group: SerPort 30

3.573 OnSysTick

Procedure *OnSysTick [(SaveAllRegs)];*
Callback. Wird bei jedem SysTick (Timer Interrupt) aufgerufen.

Group: System 7

3.574 OnTickTimer

Procedure *OnTickTimer [(SaveAllRegs)];*
Callback. Wird bei jedem TimerTick aufgerufen.

Group: System 7

3.575 OnTINA_SS

Procedure *OnTINA_SS;*
Callback. Zur Steuerung des SPI SS-Pin durch die Applikation.

Group: System 7

3.576 Ord

Function *Ord (ch : char) : byte;*
Liefert Ordnungszahl eines Zeichens/Characters.

Group: Maths 5

3.577 PadLeft

Function *PadLeft (**const** st : string; len : byte [;pad : char]) : string;*
Fügt führende Zeichen vor den String ein.

Group: Strings 6

3.578 PadRight

Function *PadRight (**const** st : string; len : byte [;pad : char]) : string;*
Hängt Zeichen an den String an.

Group: Strings 6

3.579 Parity

Function *Parity (**const** b : byte|char) : boolean;*
Ergibt die Parität eines Bytes oder Chars. odd = true.

Group: Maths 5

3.580 PipeFlush

Procedure *PipeFlush (p : pipe);*
Eine Pipe komplett leeren. Auch RxBuffer und RxBuffer1, -2,- 3.

Group: Pipes 27

3.581 PipeFull

Function *PipeFull (p : pipe) : boolean;*
Der Voll-Status einer Pipe wird abgefragt.

Group: Pipes 27

3.582 PipeRecv

Function *PipeRecv (p : pipe) : type;*
Ein Argument aus einer Pipe abholen (entfernen).

Group: Pipes 27

3.583 PipeRecv_ND

Function *PipeRecv_ND (pipe1 : pipe) : type;*
Gestattet das Auslesen einer Pipe, ohne den Inhalt selbst zu verändern.

Group: Pipes 27

3.584 PipeSend

Function *PipeSend (p : pipe; v : type) : boolean;*
Ein Argument in eine Pipe einfügen (anhängen).

Group: Pipes 27

3.585 PipeStat

Function *PipeStat (p : pipe) : byte;*
Der Inhalt bzw. Parameterzahl einer Pipe wird abgefragt.

Group: Pipes 27

3.586 PopAllRegs

Procedure *PopAllRegs;*
Spezielle Register Rück-Sicherung für verschachtelte (nested) Interrupts.

Group: System 7

3.587 PopRegs

Procedure *PopRegs; // working registers from stack*
vereinfachte Version von PopAllRegs.Kann jederzeit zusammen mit PushRegs in Interrupts eingesetzt werden wenn in Interrupts nur die 4 wesentlichen Register (ACCA, ACCB etc) gesichert wurden

Group: System 7

3.588 Pos

Function *Pos (a : char; s : string) : byte;*
Gibt die Position eines Zeichens in einem String zurück.

Group: Strings 6

3.589 PosN

Function *PosN (a : char; s : string; start : byte) : byte;*
Gibt die Position des Characters im String zurück. Gesucht wird ab der Stelle *start* im String.

Group: Strings 6

3.590 Pow

Function *Pow (x,y : float) : float;*
Liefert das Ergebnis von x hoch y.

Group: Maths 5

3.591 Pow10

Function *Pow10 (x : float) : float;*
Liefert das Ergebnis von 10 hoch x.

Group: Maths 5

3.592 PowerSave

Procedure *PowerSave (**const** mode : byte; **const** ticks : word);*
Legt die CPU n SysTicks schlafen.

Group: System 7

3.593 Pred

Function *Pred (x : type) : type;*
Gibt den nächst kleineren Wert einer Variablen zurück.

Group: Maths 5

3.594 PresetAVfilter

Procedure *PresetAVfilter (**var** Filter : AVfilter; val : type);*
Besetzt das komplette Filter mit "val".

Group: Diverse 4

3.595 Priority

Procedure *Priority (p : process|task; prio : byte);*
Setzt die Priorität eines Tasks oder Prozesses.

Group: MultiTasking 5

3.596 ProcWaitFlag

Function *ProcWaitFlag (Flag : var [; timeout : word]) : boolean;*
Flag kann eine beliebige Variable sein.
Ein Prozess/Task schaltet sich inaktiv, bis das Flag <> 0 ist.

Group: MultiTasking 5

3.597 PulseCountClear

Function *PulseCountClear;*
Stopt den internen Zähler und setzt ihn auf 0.

Group: Pulse Counter 27

3.598 PulseCountClear2

Function *PulseCountClear2;*
Stopt den internen Zähler und setzt ihn auf 0.

Group: Pulse Counter 27

3.599 PulseCountStart

Function *PulseCountStart;*
Startet den Zähler nach Programmstart, PulseCountStop oder PulseCountClear

Group: Pulse Counter 27

3.600 PulseCountStart2

Function *PulseCountStart2*
Startet den Zähler nach Programmstart, PulseCountStop2 oder PulseCountClear2

Group: Pulse Counter 27

3.601 PulseCountStop

Function *PulseCountStop;*
Stopt den Zählvorgang ohne den internen Zähler zu verändern.

Group: Pulse Counter 27

3.602 PulseCountStop2

Function *PulseCountStop2;*
Stopt den Zählvorgang ohne den internen Zähler zu verändern.

Group: Pulse Counter 27

3.603 PushAllRegs

Procedure *PushAllRegs;*
Spezielle Register Sicherung für verschachtelte (nested) Interrupts.

Group: System 7

3.604 PushRegs

Procedure *PushRegs; // working registers to stack*
vereinfachte Version von PushAllRegs.Kann jederzeit zusammen mit PopRegs
in Interrupts eingesetzt werden wenn in Interrupts nur die 4 wesentlichen Register
(ACCA, ACCB etc) gesichert wurden

Group: System 7

3.605 RadToDeg

Function *RadToDeg (w : float) : float;*
Wandeln ein Bogenmaß in Gradmaß um.

Group: Maths 5

3.606 RaiseException

Procedure *RaiseException (num : byte);*
Löst die Exception (Ausnahme) aus. Siehe try

Group: System 7

3.607 Random

Function *Random : word;*
Liefert eine Zufallszahl vom Typ Word.

Random muss importiert werden:
From *System* ***Import*** *Random****;***

Group: System 7

3.608 RandomRange

Function *RandomRange(min, max : word) : word;*
Liefert eine Zufallszahl vom Typ Word. Das Ergebnis wird durch min und max begrenzt

Random muss importiert werden:
From *System* ***Import*** *Random****;***

Group: System 7

3.609 Read

Function *Read (p : Function[; **var** st : string[; count : byte|limiter : char]]);*
Einlesen eines Zeichens oder Strings.

Group: Diverse 4

3.610 ReadKey

Function *ReadKey (**const** key : Keys) : boolean;*
Der aktuelle Status einer Taste wird zurückgegeben.

Group: KeyBoard 4x4 16

3.611 ReadKey8

Function *ReadKey8 (**const** key : Keys) : boolean;*
Der aktuelle Status einer Taste wird zurückgegeben.

Group: KeyBoard 8x8 17

3.612 ReadKeyBoard

Function *ReadKeyBoard : BitSet of Keys;*
Der aktuelle Status des Keyboards wird zurückgegeben.

Group: KeyBoard 4x4 16

3.613 ReadKeyBoard8

unction *ReadKeyBoard8 : BitSet of Keys;*
Der aktuelle Status des Keyboards wird zurückgegeben.

Group: KeyBoard 8x8 17

3.614 ReadLn

Procedure *ReadLn (DeviceFunc : function;* ***var*** *str : string);*
Liest einen String vom Device bis CRLF erscheint.

Group: Diverse 4

3.615 RecvRC5

Function *RecvRC5 (****var*** *rxAdr : byte;* ***var*** *rxCmd : byte) : boolean;*
Empfängt ein RC5 Telegramm.

Group: `RC5 Driver` 28

3.616 ResetProcess

Procedure *ResetProcess (P : Name | i : ID);*
Initialisiert einen Prozess komplett neu und setzt ihn auf suspended.

Group: MultiTasking 5

3.617 ResetSysTimer

Procedure *ResetSysTimer (tm : tSysTimer);*
Setzt einen SysTimer auf den Wert 0.

Group: Diverse 4

3.618 RestoreInts

Procedure *RestoreInts;*
Bedingte Freigabe der Interrupts abhängig vom entsprechenden System Flag.

Group: System 7

3.619 Resume

Procedure *Resume (p : process|task);*
Ein deaktivierter Prozess/Task wird aktiviert.

Group: MultiTasking 5

3.620 ResumeAll

Procedure *ResumeAll (Processes, Tasks);*
Procedure *ResumeAll (Processes);*
Procedure *ResumeAll (Tasks);*
Alle deaktivierte Prozesse und/oder Task werden aktiviert.

Group: MultiTasking 5

3.621 RotatePntI

Procedure *RotatePntI (angle, XPo, YPo : integer;* ***var*** *XPd, YPd : integer);*
Der Punkt(XPo, YPo) wird mit dem Winkel angle rotiert (Grad).

Group: Maths 5

3.622 Round

Function *Round (f : float|fix64) : integer; {Byte,Word,LongInt,LongWord}*
Rundet ein Float, Fix64 zu einem Integer, Byte, Word etc.

Group: Maths 5

3.623 RTCalarm

Procedure *RTCalarm;*
CallBack. Wird bei einem compare-match aufgerufen.

Group: RTC Driver 29

3.624 RTCalarm_Date

Procedure *RTCalarm_Date (year, month, day : byte);*
Setzt die Datum-Compare Register von RTCalarm.

Group: RTC Driver 29

3.625 RTCalarm_Start

Procedure *RTCalarm_Start (mode : byte);*
Startet oder stoppt RTCalarm.

Group: RTC Driver 29

3.626 RTCalarm_Stop

Procedure *RTCalarm_Stop;*
Stoppt RTCalarm.

Group: RTC Driver 29

3.627 RTCalarm_Time

Procedure *RTCalarm_Time (hour, min, sec : byte);*
Setzt die Zeit-Compare Register von RTCalarm.

Group: RTC Driver 29

3.628 RTCgetDay

Function *RTCgetDay : byte;*
Liest das Tages Register.

Group: RTC Driver 29

3.629 RTCgetHour

Function *RTCgetHour : byte;*
Liest das Stunden Register.

Group: RTC Driver 29

3.630 RTCgetMinute

Function *RTCgetMinute : byte;*
Liest das Minuten Register.

Group: RTC Driver 29

3.631 RTCgetMonth

Function *RTCgetMonth : byte;*
Liest das Monats Register.

Group: RTC Driver 29

3.632 RTCgetSecond

Function *RTCgetSecond : byte;*
Liest das Sekunden Register.

Group: RTC Driver 29

3.633 RTCgetWeekDay

Function *RTCgetWeekDay : byte; {0 = sunday}*
Liest das Wochentags Register.

Group: RTC Driver 29

3.634 RTCgetYear

Function *RTCgetYear : byte;*
Liest das Jahres Register.

Group: RTC Driver 29

3.635 RTCsetDay

Procedure *RTCsetDay (day : byte);*
Setzt das Tages Register mit dem Wert „day".

Group: RTC Driver 29

3.636 RTCsetHour

Procedure *RTCsetHour (hour : byte);*
Setzt das Stunden Register mit dem Wert „hour".

Group: RTC Driver 29

3.637 RTCsetMinute

Procedure *RTCsetMinute (min : byte);*
Setzt das Minuten Register mit dem Wert „min".

Group: RTC Driver 29

3.638 RTCsetMonth

Procedure *RTCsetMonth (month : byte);*
Setzt das Monats Register mit dem Wert „month".

Group: RTC Driver 29

3.639 RTCsetSecond

Procedure *RTCsetSecond (sec : byte);*
Setzt das Sekunden Register mit dem Wert „sec".

Group: RTC Driver 29

3.640 RTCsetWeekDay

Procedure *RTCsetWeekDay (wday : byte); {0 = sunday}*
Setzt das Wochentag Register mit dem Wert „wday".

Group: RTC Driver 29

3.641 RTCsetYear

Procedure *RTCsetYear (year : byte);*
Setzt das Jahres Register mit dem Wert „year".

Group: RTC Driver 29

3.642 RTCtickHour

Procedure *RTCtickHour;*
CallBack. Wird bei jedem Stunden Übertrag aufgerufen.

Group: RTC Driver 29

3.643 RTCtickMinute

Procedure *RTCtickMinute;*
CallBack. Wird bei jedem Minuten Übertrag aufgerufen.

Group: RTC Driver 29

3.644 RTCtickSecond

Procedure *RTCtickSecond;*
CallBack. Wird bei jedem Sekunden Übertrag aufgerufen.

Group: RTC Driver 29

3.645 RTCtimer

Procedure *RTCtimer;*
Procedure *RTCtimer (chan : byte);*
CallBack. Wird bei Downcounter = 0 aufgerufen.

Group: RTC Driver 29

3.646 RTCtimer_Load

Procedure *RTCtimer_Load ([chan : byte;] seconds : word[longword]);*
Setzt die Zeit-Compare Register von RTCtimerX. Compare wird gestoppt.

Group: RTC Driver 29

3.647 RTCtimer_Start

Procedure *RTCtimer_Start [(chan : byte)];*
Startet den RTCtimer.

Group: RTC Driver 29

3.648 RTCtimer_Stop

Procedure *RTCtimer_Stop[(chan : byte)];*
Stoppt den RTCtimer.

Group: RTC Driver 29

3.649 RunErr

Function *RunErr : boolean;*
Gibt den aktuellen Laufzeit Fehler zurück.

Group: System 7

3.650 RunTimeErr

procedure *RunTimeErr;*
CallBack. Wird aufgerufen wenn Fehler aufgetreten sind.

Group: System 7

3.651 Schedule

Procedure *Schedule;*
Der Prozess/Task wird an dieser Stelle abgebrochen.

Group: MultiTasking 5

3.652 SchedulerOff

Procedure *SchedulerOff;*
Der Prozess-Scheduler wird angehalten.

Group: MultiTasking 5

3.653 SchedulerOn

Procedure *SchedulerOn;*
Der Prozess-Scheduler wird wieder freigegeben.

Group: MultiTasking 5

3.654 sDelay

Procedure *sDelay (d : byte);*
Software Delay in CPU Zyklen.

Group: System 7

3.655 SemaStat

Function *SemaStat (s : semaphore) : byte;*
Der Inhalt der Semaphore wird abgefragt.

Group: System 7

3.656 SendRC5

Procedure *SendRC5 (**const** txAdr : byte; **const** txCmd : byte);*
Schickt die beiden Bytes über den IR-Sender ab.

Group: RC5 Driver 28

3.657 Ser_Enable

Procedure *Ser_Enable1 (**const** ena : boolean);*
Procedure *Ser_Enable2 (**const** ena : boolean);*
. . .

XMega:
Procedure *Ser_EnableC0 (**const** ena : boolean);*
Procedure *Ser_EnableC1 (**const** ena : boolean);*
Procedure *Ser_EnableD0 (**const** ena : boolean);*
. . .

Wird zur Steuerung des RS485 Leitungstreibers benutzt.

Bemerkung: ***Procedure*** *Ser_Enable (...)*
wurde durch ***Procedure*** *Ser_Enable1 (...)* ersetzt, existiert aber weiterhin.

Group: SerPorts 30

3.658 SerBaud

Procedure *SerBaud1 (**var** baud : word);*
Procedure *SerBaud2 (**var** baud : word);*
. . .

XMega:
Procedure *SetSerBaud (UsartC0, **var** baud : word);*
Procedure *SetSerBaud (UsartC1, **var** baud : word);*
Procedure *SetSerBaud (UsartD0, **var** baud : word);*
. . .

Stellt die Baudrate für SerPorts ein.

Bemerkung: ***Procedure*** *SerBaud (...)*
wurde durch ***Procedure*** *SerBaud1* ersetzt, existiert aber weiterhin.

Group: SerPorts

3.659 SetSerBaud

XMega:
Procedure *SetSerBaud (UsartC0,* ***var*** *baud : word);*
Procedure *SetSerBaud (UsartC1,* ***var*** *baud : word);*
Procedure *SetSerBaud (UsartD0,* ***var*** *baud : word);*
. . .

Stellt die Baudrate für SerPorts ein.

Group: SerPorts

3.660 SerDataBits

Procedure *SerDataBits1 (bits : tDataBits);*
Procedure *SerDataBits2 (bits : tDataBits);*
. . .

XMega:
Procedure *SerDataBitsC0 (bits : tDataBits);*
Procedure *SerDataBitsC1 (bits : tDataBits);*
Procedure *SerDataBitsD0 (bits : tDataBits);*
. . .

Setzt die Anzahl der Data bits.

Bemerkung: ***Procedure*** *SerDataBits (...)*
wurde durch ***Procedure*** *SerDataBits1* ersetzt, existiert aber weiterhin.

Group: SerPorts

3.661 SerInp

Function *SerInp1 : byte|char;*
Function *SerInp2 : byte|char;*
. . .

XMega:
Function *SerInpC0 : byte|char;*
Function *SerInpC1 : byte|char;*
Function *SerInpD0 : byte|char;*
. . .

Liest die serielle Schnittstelle bzw. Buffer.

Bemerkung: ***Function*** *SerInp*
wurde durch ***Function*** *SerInp1* ersetzt, existiert aber weiterhin.

Group: SerPorts

3.662 SerInp_TO

Function *SerInp_TO1 (**var** rx : char|byte; **const** timeout : byte) : boolean;*
Function *SerInp_TO2 (**var** rx : char|byte; **const** timeout : byte) : boolean;*
. . .

XMega:
Function *SerInp_TOC0 (**var** rx : char|byte; **const** timeout : byte) : boolean;*
Function *SerInp_TOC1 (**var** rx : char|byte; **const** timeout : byte) : boolean;*
Function *SerInp_TOD0 (**var** rx : char|byte; **const** timeout : byte) : boolean;*
. . .

Liest die serielle Schnittstelle bzw. Buffer.mit Timeout.

Bemerkung: ***Function*** *SerInp_TO (...)*
wurde durch ***Function*** *SerInp_TO1* ersetzt, existiert aber weiterhin.

Group: SerPorts

3.663 SerInpBlock

Procedure *SerInpBlock1 (**var** location: type);*
Procedure *SerInpBlock2 (**var** location: type);*
. . .

XMega:
Procedure *SerInpBlockC0 (**var** location: type);*
Procedure *SerInpBlockC1 (**var** location: type);*
Procedure *SerInpBlockD0 (**var** location: type);*
. . .

Empfängt einen Datenblock.

Bemerkung: ***Procedure*** *SerInpBlock (...)*
wurde durch ***Procedure*** *SerInpBlock1 (...)* ersetzt, existiert aber weiterhin.

Group: SerPorts

3.664 SerInpBlock_P

Procedure *SerInpBlock1_P (p : pointer; len : word);*
Procedure *SerInpBlock2_P (p : pointer; len : word);*
. . .

XMega:
Procedure *SerInpBlockC0_P (p : pointer; len : word);*
Procedure *SerInpBlockC1_P (p : pointer; len : word);*
Procedure *SerInpBlockD0_P (p : pointer; len : word);*
. . .

Empfängt einen Datenblock mit variabler Länge.

Bemerkung: ***Procedure*** *SerInpBlock_P (...)*
wurde durch ***Procedure*** *SerInpBlock1_P (...)* ersetzt, existiert aber weiterhin.

Group: SerPorts 30

3.665 SerInpBlock_TO

Function *SerInpBlock_TO1 (**var** location : type; **const** TimeOut : byte) : boolean;*
Function *SerInpBlock_TO2 (**var** location : type; **const** TimeOut : byte) : boolean;*
. . .

XMega:
Function *SerInpBlock_TOC0 (**var** location : type; **const** TimeOut : byte) : boolean;*
Function *SerInpBlock_TOC1 (**var** location : type; **const** TimeOut : byte) : boolean;*
Function *SerInpBlock_TOD0 (**var** location : type; **const** TimeOut : byte) : boolean;*
. . .

Empfängt einen Datenblock mit Timeout.

Bemerkung: ***Function*** *SerInpBlock_TO (...)*
wurde durch ***Function*** *SerInpBlock_TO1 (...)* ersetzt, existiert aber weiterhin.

Group: SerPorts 30

3.666 SerInpBlockP_TO

Function *SerInpBlockP_TO1 (p : pointer; len : word; to : byte) : boolean;*
Function *SerInpBlockP_TO2 (p : pointer; len : word; to : byte) : boolean;*
. . .

XMega:
Function *SerInpBlockP_TOC0 (p : pointer; len : word; to : byte) : boolean;*
Function *SerInpBlockP_TOC1 (p : pointer; len : word; to : byte) : boolean;*
Function *SerInpBlockP_TOD0 (p : pointer; len : word; to : byte) : boolean;*
. . .

Empfängt einen Datenblock variabler Länge mit Timeout.

Bemerkung: ***Function*** *SerInpBlockP_TO (...)*
wurde durch ***Function*** *SerInpBlockP_TO1 (...)* ersetzt, existiert aber weiterhin.

Group: SerPorts 30

3.667 SerInpSLIP

Function *SerInpSLIP1 (dst : pointer; tmo : byte; count : word) : word;*
Function *SerInpSLIP2 (dst : pointer; tmo : byte; count : word) : word;*
. . .

XMega:
Function *SerInpSLIPC0(dst : pointer; tmo : byte; count : word) : word; // UARTC0*
Function *SerInpSLIPC1(dst : pointer; tmo : byte; count : word) : word; // UARTC1*
Function *SerInpSLIPD0(dst : pointer; tmo : byte; count : word) : word; // UARTD0*
. . .

Empfängt ein SLIP Packet.

Bemerkung: ***Function*** *SerInpSLIP (...)*
wurde durch ***Function*** *SerInpSLIP1 (...)* ersetzt, existiert aber weiterhin.

Group: SerPorts

3.668 SerOut

Procedure *SerOut1 (ch : byte|char);*
Procedure *SerOut2 (ch : byte|char);*
. . .

XMega:
Procedure *SerOutC0 (ch : byte|char);*
Procedure *SerOutC1 (ch : byte|char);*
Procedure *SerOutD0 (ch : byte|char);*
. . .

Schreibt ein Zeichen in den Sendebuffer.

Bemerkung: ***Procedure*** *SerOut (...)*
wurde durch ***Procedure*** *SerOut1 (...)* ersetzt, existiert aber weiterhin.

Group: SerPorts

3.669 SerOutBlock

Procedure *SerOutBlock1 (**const** location: type);*
Procedure *SerOutBlock2 (**const** location: type);*
. . .

XMega:
Procedure *SerOutBlockC0 (**const** location: type);*
Procedure *SerOutBlockC1 (**const** location: type);*
Procedure *SerOutBlockD0 (**const** location: type);*
. . .

Sendet einen Datenblock.

Bemerkung: ***Procedure*** *SerOutBlock (...)*
wurde durch ***Procedure*** *SerOutBlock1 (...)* ersetzt, existiert aber weiterhin.

Group: SerPorts

3.670 SerOutBlock_P

Procedure *SerOutBlock1_P (p : pointer; count : word);*
Procedure *SerOutBlock2_P (p : pointer; count : word);*
. . .

XMega:
Procedure *SerOutBlockC0_P (p : pointer; count : word);*
Procedure *SerOutBlockC1_P (p : pointer; count : word);*
Procedure *SerOutBlockD0_P (p : pointer; count : word);*

. . .

Sendet einen Datenblock variabler Länge.

Bemerkung: ***Procedure*** *SerOutBlock_P*
wurde durch ***Procedure*** *SerOutBlock1_P (...)* ersetzt, existiert aber weiterhin.

Group: SerPorts 30

3.671 SerOutSLIP

Procedure *SerOutSLIP1 (src : pointer; count : word);*
Procedure *SerOutSLIP2 (src : pointer; count : word);*
. . .

XMega:
Procedure *SerOutSLIPC0(src : pointer; count : word); // UARTC0*
Procedure *SerOutSLIPC1(src : pointer; count : word); // UARTC1*
Procedure *SerOutSLIPD0(src : pointer; count : word); // UARTD0*
. . .

Bildet und sendet ein SLIP Packet.

Bemerkung: ***Procedure*** *SerOutSLIP (...)*
wurde durch ***Procedure*** *SerOutSLIP1 (...)* ersetzt, existiert aber weiterhin.

Group: SerPorts 30

3.672 SerParity

Procedure *SerParity1 (par : tParity);*
Procedure *SerParity2 (par : tParity);*
. . .

XMega:
Procedure *SerParityC0 (par : tParity);*
Procedure *SerParityC1 (par : tParity);*
Procedure *SerParityD0 (par : tParity);*
. . .

Bestimmt das Parity Bit, eben, odd, none.

Bemerkung: ***Procedure*** *SerParity (...)*
wurde durch ***Procedure*** *SerParity1 (...)* ersetzt, existiert aber weiterhin.

Group: SerPorts 30

3.673 SerPort_Send

Procedure *SerPort_Send1;*
Procedure *SerPort_Send2;*
. . .

XMega:

Procedure *SerPort_SendC0;*
Procedure *SerPort_SendC1;*
Procedure *SerPort_SendD0;*
. . .

Freigabe bzw. Aufhebung der Sperrung (DTR).

Bemerkung: ***Procedure*** *SerPort_Send*
wurde durch ***Procedure*** *SerPort_Send1* ersetzt, existiert aber weiterhin.

Group: SerPorts 30

3.674 SerStat

Function *SerStat1 : boolean;*
Function *SerStat2 : boolean;*
. . .

XMega:
Function *SerStatC0 : boolean;*
Function *SerStatC1 : boolean;*
Function *SerStatD0 : boolean;*
. . .

Liefert TRUE zurück, wenn ein Zeichen vorhanden ist.

Bemerkung: ***Function*** *SerStat*
wurde durch ***Function*** *SerStat1* ersetzt, existiert aber weiterhin.

Group: SerPorts 30

3.675 SerStopBits

Procedure *SerStopBits1 (bits : tStopBits);*
Procedure *SerStopBits2 (bits : tStopBits);*
. . .

XMega:
Procedure *SerStopBitsC0 (bits : tStopBits);*
Procedure *SerStopBitsC1 (bits : tStopBits);*
Procedure *SerStopBitsD0 (bits : tStopBits);*
. . .

Setzt die Anzahl der Stopbits.

Bemerkung: ***Procedure*** *SerStopBits (...)*
wurde durch ***Procedure*** *SerStopBits1 (...)* ersetzt, existiert aber weiterhin.

Group: SerPorts 30

3.676 SetAdcFixed

Procedure *SetAdcFixed (fix : boolean; chan : byte);*
Konvertiert kontinuierlich einen bestimmter Kanal (fix = true).

Achtung: der AVRco zählt die Kanäle 1, 2, 3, ... (entsprechend 0, 1, 2, ... beim Controller)

Group: ADC 8

3.677 SetAVfilter

Function *SetAVfilter (**var** Filter : AVfilter; val : type) : type;*
Ersetzt den ältesten Eintrag durch den Wert "val" und liefert den neu gerechneten Mittelwert zurück.

Group: Diverse 4

3.678 SetBit

Procedure *SetBit (BitType, boolean);*
Setzt oder setzt zurück das Bit.

Group: System 7

3.679 SetDacA

Function *SetDacA_chan0 (val : word);*
Function *SetDacA_chan1 (val : word);*
Schreibt einen neuen Ausgabewert (unteren 12bit) in das DAC Register.

Group: DA Converter 10

3.680 SetDacB

Function *SetDacB_chan0 (val : word);*
Function *SetDacB_chan1 (val : word);*
Schreibt einen neuen Ausgabewert (unteren 12bit) in das DAC Register.

Group: DA Converter 10

3.681 SetDeviceLock

Function *SetDeviceLock (d : DeviceLock) : boolean;*
Kehrt mit einem true zurück, wenn das Device frei war.

Group: Diverse 4

3.682 SetFreqCountMode

Procedure *SetFreqCountMode (mode : tFreqCountMode);*
Der Parameter *mode* ist die gewünschte Auflösung.

Group: Frequency Counter/Timer 11

3.683 SetFreqCountMode2

Procedure *SetFreqCountMode2(mode : tFreqCountMode);*
Der Parameter *mode* ist die gewünschte Auflösung.

Group: Frequency Counter/Timer 11

3.684 SetIncrementVal

Procedure *SetIncrementVal (val : integer [longint]);*
Setzt den absoluten internen Zähler auf den Wert „val".

Group: Increment Counter 15

3.685 SetIncrVal4

Procedure *SetIncrVal4 (chan : byte; val : integer [longint]);*
Setzt den absoluten internen Zähler *chan* auf den Wert „val".

Group: Increment Counter 4chan 15

3.686 SetLength

Procedure *SetLength (st : string; len : byte);*
Verkürzt/verlängert den String auf die angegebene Länge.

Group: Strings 6

3.687 SetMSPIclkPha

Procedure *SetMSPIclkPha0 (phase : byte);*
Procedure *SetMSPIclkPha1 (phase : byte);*
. . .
Group: ATMega MSPI HardwareDriver 35

XMega:
Procedure *SetMSPIclkPha_C0 (phase : byte);*
Procedure *SetMSPIclkPha_C1 (phase : byte);*
Procedure *SetMSPIclkPha_D0 (phase : byte);*
. . .
Group: XMega MSPI HardwareDriver 36

Bestimmt die Clock Phase. *phase* = 0/1.

3.688 SetMSPIclkPol

Procedure *SetMSPIclkPol0 (pol : byte);*
Procedure *SetMSPIclkPol1 (pol : byte);*
. . .
Group: ATMega MSPI HardwareDriver 35

XMega:
Procedure *SetMSPIclkPol_C0 (pol : byte);*
Procedure *SetMSPIclkPol_C1 (pol : byte);*
Procedure *SetMSPIclkPol_D0 (pol : byte);*
. . .
Group: XMega MSPI HardwareDriver 36

Bestimmt die Clock Polarität. *pol* = 0/1

3.689 SetMSPImode

Procedure *SetMSPI0mode (mode : byte);*
Procedure *SetMSPI1mode (mode : byte);*
. . .
Group: ATMega MSPI HardwareDriver 35

XMega:
Procedure *SetMSPImode_C0 (mode : byte);*
Procedure *SetMSPImode_C1 (mode : byte);*
Procedure *SetMSPImode_D0 (mode : byte);*
. . .
Group: XMega MSPI HardwareDriver 36

Bestimmt den Mode von MSPI 1. *mode* = 0, 1, 2, oder 3.

3.690 SetMSPIorder

Procedure *SetMSPIorder0 (msb : boolean);*
Procedure *SetMSPIorder1 (msb : boolean);*
. . .
Group: ATMega MSPI HardwareDriver 35

XMega:
Procedure *SetMSPIorder_C0 (msb : boolean);*
Procedure *SetMSPIorder_C1 (msb : boolean);*
Procedure *SetMSPIorder_D0 (msb : boolean);*
. . .
Group: XMega MSPI HardwareDriver 36

Bestimmt die Bit Reihenfolge, msb oder lsb.

3.691 SetMSPIpresc

Procedure *SetMSPIpresc0 (presc : byte);*
Procedure *SetMSPIpresc1 (presc : byte);*
. . .
Group: ATMega MSPI HardwareDriver 35

XMega:
Procedure *SetMSPIpresc_C0 (presc : boolean);*
Procedure *SetMSPIpresc_C1 (presc : boolean);*
Procedure *SetMSPIpresc_D0 (presc : boolean);*
. . .
Group: XMega MSPI HardwareDriver 36

Bestimmt den Clock Vorteiler. *presc* = 0..255

3.692 SetPWM

procedure *SetPWM_C0A (pw : byte|word);*
procedure *SetPWM_C0B (pw : byte|word);*
procedure *SetPWM_C0C (pw : byte|word);*
. . .

nur XMega
Gibt einen neuen Wert auf dem jeweiligen PWM Port aus.
pw wird durch das entsprechende Define *PWMres_xxx* bestimmt (>8 Bit Word, sonst Byte)

Group: PWM Port 27

3.693 SetSema

Procedure *SetSema (sema : semaphore; v : byte);*
Setzt eine Semaphore.

Group: System 7

3.694 SetServoChan

Procedure *SetServoChan (chan : byte; pulse : integer);*
Verändert der Servo Position.

Group: Servo Driver 32

3.695 SetServoOffs

Procedure *SetServoOffs (chan : byte; offs : integer);*
Verändert die Servo 0-Position.

Group: Servo Driver 32

3.696 SetSPIclkPha

Procedure *SetSPIclkPha (phase : byte);*

XMega:
Procedure *SetSPIclkPhaC (phase : byte);*
Procedure *SetSPIclkPhaD (phase : byte);*
Procedure *SetSPIclkPhaE (phase : byte);*
Procedure *SetSPIclkPhaF (phase : byte);*

Bestimmt die Clock Phase. *phase* = 0/1.

Group: SPI HardwareDriver 37

3.697 SetSPIclkPol

Procedure *SetSPIclkPol (pol : byte);*

XMega:
Procedure *SetSPIclkPolC (pol : byte);*
Procedure *SetSPIclkPolD (pol : byte);*
Procedure *SetSPIclkPolE (pol : byte);*
Procedure *SetSPIclkPolF (pol : byte);*

Bestimmt die Clock Polarität. 0/1

Group: SPI HardwareDriver 37

3.698 SetSPIdoubleSpeed

Procedure *SetSPIdoubleSpeed (ds : boolean);*

XMega:
Procedure *SetSPIdoubleSpeedC (ds : boolean);*
Procedure *SetSPIdoubleSpeedD (ds : boolean);*
Procedure *SetSPIdoubleSpeedE (ds : boolean);*
Procedure *SetSPIdoubleSpeedF (ds : boolean);*

Schaltet die SPI Datenrate zwischen doppelt und einfach.

Group: SPI HardwareDriver 37

3.699 SetSPImode

Procedure *SetSPImode (mode : byte);*

XMega:
Procedure *SetSPImodeC (mode : byte);*
Procedure *SetSPImodeD (mode : byte);*
Procedure *SetSPImodeE (mode : byte);*
Procedure *SetSPImodeF (mode : byte);*

Bestimmt den SPI Mode. *mode* = 0, 1, 2 oder 3

Group: SPI HardwareDriver 37

3.700 SetSPIorder

Procedure *SetSPIorder (msb : boolean);*

XMega:
Procedure *SetSPIorderC (msb : boolean);*
Procedure *SetSPIorderD (msb : boolean);*
Procedure *SetSPIorderE (msb : boolean);*
Procedure *SetSPIorderF (msb : boolean);*

Bestimmt die Bit Reihenfolge, msb oder lsb.

Group: SPI HardwareDriver 37

3.701 SetSPIpresc

Procedure *SetSPIpresc (presc : byte);*

XMega:
Procedure *SetSPIprescC (presc : byte);*
Procedure *SetSPIprescD (presc : byte);*
Procedure *SetSPIprescE (presc : byte);*
Procedure *SetSPIprescF (presc : byte);*

Bestimmt den Clock Vorteiler. *presc* = 0..3 -> 4/16/64/128

Group: SPI HardwareDriver 37

3.702 SetSysBlinkTimer

Procedure *SetSysBlinkTimer (t : byte);*
Ändert die Blinkrate zur Laufzeit. t = 2..255 in SysTicks.

Group: SysLeds 40

3.703 SetSysTimer

Procedure *SetSysTimer (tm : SysTimer; time : byte|word);*
Timer setzen, Parameter in SysTicks.

Group: Diverse 4

3.704 SetSysTimerM

Procedure *SetSysTimerM (tm : SysTimer; time : byte|word);*
Timer setzen, Parameter in Millisekunden.

Group: Diverse 4

3.705 SetTable

Procedure *SetTable (t : Table; index : byte; new : type);*
Verändert ein Mitglied einer LookUp-Table.

Group: System 7

3.706 SetTWImode

Procedure *SetTWImode (**const** twimode : tTWInetmode);*
Schaltet zwischen Master und Slave um.

Group: TWI Network 44

3.707 SetTWInodeAddr

Procedure *SetTWInodeAddr (sAddr : byte);*
Setzt die Netzwerk Adresse des Masters oder Slaves.

Group: TWI Network 44

3.708 SetVectTabBoot

Procedure *SetVectTabBoot (boot : boolean);*
Umschalten der Vektor Tabellen (IVSEL, MCUCR/GICR).

Group: System 7

3.709 Sgn

Function *Sgn (**const** num : integer, int8, longint, word64, int64, fix64, float) : type;*
Ist das Argument > 0 ergibt die Funktion ein '1',
ist das Argument '0' so ist das Resultat '0', ansonsten '-1'
Das Ergebnis ist vom gleichen Typ wie das Argument.

Group: Maths 5

3.710 SHT11convState

Function *SHT11convState : boolean;*
Gibt den Status der Conversion zurück.

Group: SHT11 Driver 32

3.711 SHT11getHum

Function *SHT11getHum : word;*
Liest das Messergebnis aus dem Sensor.

Group: SHT11 Driver 32

3.712 SHT11getStatus

Function *SHT11getStatus : byte;*
Liest das Status/Steuer Register des Sensors aus.

Group: SHT11 Driver

3.713 SHT11getTemp

Function *SHT11getTemp : word;*
Liest das Messergebnis aus dem Sensor.

Group: SHT11 Driver

3.714 SHT11setStatus

Procedure *SHT11setStatus (s : byte);*
Diverse Sonderfunktionen.

Group: SHT11 Driver

3.715 SHT11softReset

Procedure *SHT11softReset;*
Setzt das Status/Steuer Register auf Default Werte.

Group: SHT11 Driver

3.716 SHT11startHum

Procedure *SHT11startHum;*
Startet die Feuchte Wandlung.

Group: SHT11 Driver

3.717 SHT11startTemp

Procedure *SHT11startTemp;*
Startet die Temperatur Wandlung.

Group: SHT11 Driver

3.718 SHT11synchronize

Procedure *SHT11synchronize;*
Setzt das 2-Draht Interface zurück.

Group: SHT11 Driver

3.719 Sign

Function *Sign (**const** num : integer[int8, longint, word64, int64, fix64, float]) : boolean;*
Ist das Argument positiv ergibt die Funktion ein true, ansonsten ein false.

Group: Maths 5

3.720 Sin

Function *Sin (w : float) : float;*
Liefert den Sinus des Arguments zurück.

Group: Maths 5

3.721 SinD

Function *SinD (w : float) : float;*
Liefert den Sinus des Arguments zurück.

Group: Maths 5

3.722 SinInt

Function *SinInt (angle, v : integer) : integer;*
Liefert den Sinus des Winkels multipliziert mit dem Integer Argument.

Group: Maths 5

3.723 SinInt16

Function *SinInt16 (angle : integer) : integer; // angle in 0.1deg*
Errechnet den Sinus des Winkels, multipliziert diesen mit 10000.

Group: Maths 5

3.724 SizeOf

Function *SizeOf (x : type) : word;*
Gibt den Speicherbedarf eines Objekts in Bytes zurück.

Group: Diverse 4

3.725 Sleep

Procedure *Sleep (p : process; t : word);*
Prozess Sleep in Ticks.

Group: System 7

3.726 SLIPgetRxCount

Function *SLIPgetRxCount1 : word;*
Function *SLIPgetRxCount2 : word;*
. . .

XMega:
Function *SLIPgetRxCountC0: word;*
Function *SLIPgetRxCountC1: word;*
Function *SLIPgetRxCountD0: word;*
. . .

Abfrage Byte Count des empfangenen Packets wenn Rx Status *SLIPready.*

Group: SLIPport 33

3.727 SLIPgetRxState

Function *SLIPgetRxState1: tSLIPstate;*
Function *SLIPgetRxState2: tSLIPstate;*
. . .

XMega:
Function *SLIPgetRxStateC0: tSLIPstate;*
Function *SLIPgetRxStateC1: tSLIPstate;*
Function *SLIPgetRxStateD0: tSLIPstate;*
. . .

Abfrage Rx Status des Treibers.
Das mögliche Resultat: *SLIPready, SLIPbusy, SLIPovr, SLIPtout, SLIPfrm.*

Group: SLIPport 33

3.728 SLIPgetTxState

Function *SLIPgetTxState1 : tSLIPstate;*
Function *SLIPgetTxState2 : tSLIPstate;*
. . .

XMega:
Function *SLIPgetTxStateC0 : tSLIPstate;*
Function *SLIPgetTxStateC1 : tSLIPstate;*
Function *SLIPgetTxStateD0 : tSLIPstate;*
. . .

Abfrage Tx Status des Treibers.
Das mögliche Resultat: *SLIPready, SLIPbusy, SLIPtout.*

Group: SLIPport 33

3.729 SLIPresumeRx

Function *SLIPresumeRx1 : boolean;*
Function *SLIPresumeRx2 : boolean;*
. . .

XMega:
Function *SLIPresumeRxC0 : boolean;*
Function *SLIPresumeRxC1 : boolean;*
Function *SLIPresumeRxD0 : boolean;*
. . .

Empfang freigeben. Nach dem Programm Start und nach jedem Packet Empfang, auch fehlerhaften.
Ist der Rx Status des Treibers auf *SLIPbusy* kehrt die Funktion mit einem false zurück.

Group: SLIPport 33

3.730 SLIPrxReady

Function *SLIPrxReady1 : boolean;*
Function *SLIPrxReady2 : boolean;*
. . .

XMega:
Function *SLIPrxReadyC0 : boolean;*
Function *SLIPrxReadyC1 : boolean;*
Function *SLIPrxReadyD0 : boolean;*
. . .

ergibt ein true wenn das zuletzt empfangene Packet zum Abholen bereit ist.
Auch ein defektes Packet oder TimeOut ergibt ein true.
Der Status des Packets kann mit SLIPgetRxState1 abgefragt werden.

Group: SLIPport 33

3.731 SLIPsetMode

Function *SLIPsetMode1 : (sMode : tSLIPmode) : boolean;*
Function *SLIPsetMode2 : (sMode : tSLIPmode) : boolean;*
. . .

XMega:
Function *SLIPsetModeC0 : (sMode : tSLIPmode) : boolean;*
Function *SLIPsetModeC1 : (sMode : tSLIPmode) : boolean;*
Function *SLIPsetModeD0 : (sMode : tSLIPmode) : boolean;*
. . .

Initialisieren der Betriebsart des Treibers mittels Bitset Parameter.
Bitnamen: *slpHsk, slpChkS, slpAddr*
Es ist damit möglich die Konfiguration beliebig zu kombinieren aus Handshake, Checksumme und Adressprüfung.
Die Funktion kehrt mit einem false zurück wenn Rx oder Tx noch busy ist.

Achtung: im Handshake Betrieb arbeitet der Treiber im Halb-Duplex Mode.

D.h. solange das Packet inkl. Handshake nicht komplett abgearbeitet wurde,
darf kein weiteres Packet gesendet werden, von keinem der Beteiligten.

Group: SLIPport 33

3.732 SLIPsetRxAddr

Procedure *SLIPsetRxAddr1 (rxAddr : byte);*
Procedure *SLIPsetRxAddr2 (rxAddr : byte);*
. . .

XMega:
Procedure *SLIPsetRxAddrC0 (rxAddr : byte);*
Procedure *SLIPsetRxAddrC1 (rxAddr : byte);*
Procedure *SLIPsetRxAddrD0 (rxAddr : byte);*
. . .

Gibt die interne Rx Adresse vor. Diese wird benötigt wenn im Mode der Wert *slpAddr* aktiv ist.
Diese Adresse kann jederzeit geändert werden, ist aber bei anderen Modes ohne Bedeutung.

Group: SLIPport 33

3.733 SLIPsetRxBuffer

Function *SLIPsetRxBuffer1 (rxBuff : pointer; size : word) : boolean;*
Function *SLIPsetRxBuffer2 (rxBuff : pointer; size : word) : boolean;*
. . .

XMega:
Function *SLIPsetRxBufferC0 (rxBuff : pointer; size : word) : boolean;*
Function *SLIPsetRxBufferC1 (rxBuff : pointer; size : word) : boolean;*
Function *SLIPsetRxBufferD0 (rxBuff : pointer; size : word) : boolean;*
. . .

Gibt den Empfangs Buffer und dessen Grösse vor.
Wenn ein reinkommendes Packet grösser ist wird es verworfen und ein Overrun Error gesetzt.
Ist *slpChkS* aktiv so wird die Checksumme als letztes Byte in den Rx Buffer abgelegt.
Ist der Rx Status des Treibers auf *SLIPbusy* kehrt die Funktion mit einem false zurück.

Group: SLIPport 33

3.734 SLIPsetTimeOut

Procedure *SLIPsetTimeOut1 (TimeOut : byte);*
Procedure *SLIPsetTimeOut2 (TimeOut : byte);*
. . .

XMega:
Procedure *SLIPsetTimeOutC0 (TimeOut : byte);*
Procedure *SLIPsetTimeOutC1 (TimeOut : byte);*
Procedure *SLIPsetTimeOutD0 (TimeOut : byte);*
. . .

Gibt das Timeout für Empfangs Funktionen und das Handshake (in SysTicks) vor.
Bricht der Empfang eines Packets unerwartet ab, wartet der Treiber bis dieses Timeout abglaufen ist und setzt dann den RxStatus auf *SLIPtout*.
Beim Handshake Betrieb wird beim Senden ebenfalls mit einem Timeout abgebrochen wenn das Acknowledge des Empfängers ausbleibt.

Group: SLIPport 33

3.735 SLIPsetTxAddr

Procedure *SLIPsetTxAddr1 (txAddr : byte);*
Procedure *SLIPsetTxAddr2 (txAddr : byte);*
. . .

XMega:
Procedure *SLIPsetTxAddrC0 (txAddr : byte);*
Procedure *SLIPsetTxAddrC1 (txAddr : byte);*
Procedure *SLIPsetTxAddrD0 (txAddr : byte);*
. . .

Gibt die Ziel Tx Adresse vor. Diese wird benötigt wenn im Mode der Wert *slpAddr* aktiv ist.
Diese Adresse kann jederzeit geändert werden, ist aber bei anderen Modes ohne Bedeutung.

Group: SLIPport 33

3.736 SLIPsetTxBuffer

Function *SLIPsetTxBuffer1 (txBuff : pointer; size : word) : boolean;*
Function *SLIPsetTxBuffer2 (txBuff : pointer; size : word) : boolean;*
. . .

XMega:
Function *SLIPsetTxBufferC0 (txBuff : pointer; size : word) : boolean;*
Function *SLIPsetTxBufferC1 (txBuff : pointer; size : word) : boolean;*
Function *SLIPsetTxBufferD0 (txBuff : pointer; size : word) : boolean;*
. . .

Gibt den Sende Buffer und dessen Grösse vor. Ist der Tx Status des Treibers auf *SLIPbusy* dann kehrt die Funktion mit einem false zurück.

Group: SLIPport 33

3.737 SLIPstartTx

Function *SLIPstartTx1 : boolean;*
Function *SLIPstartTx2 : boolean;*
. . .

XMega:
Function *SLIPstartTxC0 : boolean;*
Function *SLIPstartTxC1 : boolean;*
Function *SLIPstartTxD0 : boolean;*
. . .

Startet einen Transfer. Die Daten die sich im *txBuff* befinden werden mit der Länge *size* (von SLIPsetTxBuffer) oder *count* (von SLIPstartTxC) gesendet.
Ist der Tx Status des Treibers auf *SLIPbusy* dann kehrt die Funktion mit einem false zurück.

Group: SLIPport 33

3.738 SLIPstartTxC

***Function** SLIPstartTxC1 (count : word) : boolean;*
***Function** SLIPstartTxC2 (count : word) : boolean;*
. . .

XMega:
***Function** SLIPstartTxC_C0 (count : word) : boolean;*
***Function** SLIPstartTxC_C1 (count : word) : boolean;*
***Function** SLIPstartTxC_D0 (count : word) : boolean;*
. . .

Startet einen Transfer. Die Daten die sich im *txBuff* befinden werden mit der Länge *count* gesendet.
Der Wert *count* gilt ab sofort, bis auf Widerruf (mit SLIPsetTxBuffer oder SLIPstartTxC) für alle weiteren Sende Operation.
Ist der Tx Status auf *SLIPbusy* dann kehrt die Funktion mit einem false zurück.

Group: SLIPport 33

3.739 SLIPstopRx

***Procedure** SLIPstopRx1;*
***Procedure** SLIPstopRx2;*
. . .

XMega:
***Procedure** SLIPstopRxC0;*
***Procedure** SLIPstopRxC1;*
***Procedure** SLIPstopRxD0;*
. . .

Wurde der Empfang mit SLIPresumeRx.. freigegeben, so kann er mit dieser Prozedur wieder gestoppt werden. Dieser Aufruf ist immer möglich.

Group: SLIPport 33

3.740 SLIPwasBC

***Function** SLIPwasBC1 : boolean;*
***Function** SLIPwasBC2 : boolean;*
. . .

XMega:
***Function** SLIPwasBC_C0 : boolean;*
***Function** SLIPwasBC_C1 : boolean;*
***Function** SLIPwasBC_D0 : boolean;*
. . .

Ergibt ein true wenn das zuletzt empfangene Packet ein Broadcast Packet war.

Group: SLIPport 33

3.741 SoftPWMstart

Procedure *SoftPWMstart;*
Der zugehörige Timer wird neu gestartet und alle PWM Werte werden auf Null gesetzt

Group: SoftwarePWM 34

3.742 SoftPWMstop

Procedure *SoftPWMstop;*
Der zugehörige Timer wird angehalten. Die PWM Pins werden auf idle/inaktiv gesetzt.

Group: SoftwarePWM 34

3.743 SpeechOutFlash

Procedure *SpeechOutFlash (start : pointer; count : word);*
Gibt einen Datenblock aus dem Flash aus.

Group: Speech Port 34

3.744 SpeechOutRAM

Procedure *SpeechOutRAM (start : pointer; count : word);*
Gibt einen Datenblock aus dem Ram aus.

Group: Speech Port 34

3.745 SpeechReady

Function *SpeechReady : boolean;*
Fragt ab, ob die Ausgabe beendet ist. Wird bei Doppel Pufferung benötigt.

Group: Speech Port 34

3.746 SpeechStop

Procedure *SpeechStop;*
Bricht eine laufende Ausgabe ab.

Group: Speech Port 34

3.747 SPIinOut

Procedure *SPIinOut (source, dest : pointer; count : word);*

XMega:
Procedure *SPIinOutC (source, dest : pointer; count : word);*
Procedure *SPIinOutD (source, dest : pointer; count : word);*
Procedure *SPIinOutE (source, dest : pointer; count : word);*
Procedure *SPIinOutF (source, dest : pointer; count : word);*

Liest und schreibt einen Datenblock.

Group: SPI HardwareDriver 37

3.748 SPIinOut1

Procedure *SPIinOut1 (source, dest : pointer; count : word);*
Liest und schreibt einen Datenblock in den SPI-Slave.

Group: SPI SoftDriver 38

3.749 SPIinOut2

Procedure *SPIinOut2 (source, dest : pointer; count : word);*
Liest und schreibt einen Datenblock in den SPI-Slave.

Group: SPI SoftDriver 38

3.750 SPIinOutByte

Function *SPIinOutByte (b : byte) : byte;*

XMega:
Function *SPIinOutByteC (b : byte) : byte;*
Function *SPIinOutByteD (b : byte) : byte;*
Function *SPIinOutByteE (b : byte) : byte;*
Function *SPIinOutByteF (b : byte) : byte;*

Liest und schreibt ein Byte in den SPI-Slave.

Group: SPI HardwareDriver 37

3.751 SPIinOutByte1

Function *SPIinOutByte1 (b : byte) : byte;*
Liest und Schreibt ein Byte in den SPI-Slave.

Group: SPI SoftDriver 38

3.752 SPIinOutByte2

Function *SPIinOutByte2 (b : byte) : byte;*
Liest und Schreibt ein Byte in den SPI-Slave.

Group: SPI SoftDriver 38

3.753 SPIinp

Function *SPIinp : byte;*

XMega:
Function *SPIinpC : byte;*
Function *SPIinpD : byte;*
Function *SPIinpE : byte;*
Function *SPIinpF : byte;*

Liest ein Byte aus dem RxBuffer.

Group: SPI Network 38

Procedure *SPIinp (dest : pointer; count : word);*

XMega:
Procedure *SPIinpC (dest : pointer; count : word);*
Procedure *SPIinpD (dest : pointer; count : word);*
Procedure *SPIinpE (dest : pointer; count : word);*
Procedure *SPIinpF (dest : pointer; count : word);*

Liest einen Datenblock aus dem SPI-Slave.

Group: SPI HardwareDriver 37

3.754 SPIinp1

Procedure *SPIinp1 (dest : pointer; count : word);*
Liest einen Datenblock aus dem SPI-Slave in die Stelle ***dest*** mit der Länge ***count.***

Group: SPI SoftDriver 38

3.755 SPIinp2

Procedure *SPIinp2 (dest : pointer; count : word);*
Liest einen Datenblock aus dem SPI-Slave in die Stelle ***dest*** mit der Länge ***count.***

Group: SPI SoftDriver 38

3.756 SPIinpByte

Function *SPIinpByte : byte;*

XMega:
Function *SPIinpByteC : byte;*
Function *SPIinpByteD : byte;*
Function *SPIinpByteE : byte;*
Function *SPIinpByteF : byte;*

Liest ein Byte aus dem SPI-Slave.

Group: SPI HardwareDriver 37

3.757 SPIinpByte1

Function *SPIinpByte1 : byte;*
Liest ein Byte aus dem SPI-Slave.

Group: SPI SoftDriver 38

3.758 SPIinpByte2

Function *SPIinpByte2 : byte;*
Liest ein Byte aus dem SPI-Slave.

Group: SPI SoftDriver 38

3.759 SPIinpLong

Function *SPIinpLong : longword;*

XMega:
Function *SPIinpLongC : longword;*
Function *SPIinpLongD : longword;*
Function *SPIinpLongE : longword;*
Function *SPIinpLongF : longword;*

Liest ein LongWord aus dem SPI-Slave.

Group: SPI HardwareDriver 37

3.760 SPIinpLong1

Function *SPIinpLong1 : longword;*
Liest ein LongWord aus dem SPI-Slave.

Group: SPI SoftDriver 38

3.761 SPIinpLong2

Function *SPIinpLong2 : longword;*
Liest ein LongWord aus dem SPI-Slave.

Group: SPI SoftDriver 38

3.762 SPIinpLong64

Function *SPIinpLong64C : longword;*
Function *SPIinpLong64D : longword;*
Function *SPIinpLong64E : longword;*
Function *SPIinpLong64F : longword;*

nur für **XMega**
Liest ein 64bit LongWord aus einem SPI-Slave.

Group: SPI HardwareDriver 37

3.763 SPIinpWord

Function *SPIinpWord : word;*

XMega:
Function *SPIinpWordC : word;*
Function *SPIinpWordD : word;*
Function *SPIinpWordE : word;*
Function *SPIinpWordF : word;*

Liest ein Word aus dem SPI-Slave.

Group: SPI HardwareDriver 37

3.764 SPIinpWord1

Function *SPIinpWord1 : word;*
Liest ein Word aus dem SPI-Slave.

Group: SPI SoftDriver 38

3.765 SPIinpWord2

Function *SPIinpWord2 : word;*
Liest ein Word aus dem SPI-Slave.

Group: SPI SoftDriver 38

3.766 SPIout

Procedure *SPIout (**const** b : byte);*

XMega:
Procedure *SPIoutC (**const** b : byte);*
Procedure *SPIoutD (**const** b : byte);*
Procedure *SPIoutE (**const** b : byte);*
Procedure *SPIoutF (**const** b : byte);*

Schreibt ein Byte in den TxBuffer.

Group: SPI Network 38

Function *SPIout (source : pointer; count : word) : boolean;*

XMega:
Function *SPIoutC (source : pointer; count : word) : boolean;*
Function *SPIoutD (source : pointer; count : word) : boolean;*
Function *SPIoutE (source : pointer; count : word) : boolean;*
Function *SPIoutF (source : pointer; count : word) : boolean;*

Schreibt ein Byte in den SPI-Slave.

Group: SPI HardwareDriver 37

3.767 SPIout1

Procedure *SPIout1 (source : pointer; count : word);*
Schreibt einen Datenblock in den SPI-Slave.

Group: SPI SoftDriver 38

3.768 SPIout2

Procedure *SPIout2 (source : pointer; count : word);*
Schreibt einen Datenblock in den SPI-Slave.

Group: SPI SoftDriver 38

3.769 SPIoutByte

Procedure *SPIoutByte (b : byte);*

XMega:
Procedure *SPIoutByteC (b : byte);*
Procedure *SPIoutByteD (b : byte);*
Procedure *SPIoutByteE (b : byte);*
Procedure *SPIoutByteF (b : byte);*

Schreibt ein Byte in den SPI-Slave.

Group: SPI HardwareDriver 37

3.770 SPIoutByte1

Procedure *SPIoutByte1 (b : byte);*
Schreibt ein Byte in den SPI-Slave.

Group: SPI SoftDriver 38

3.771 SPIoutByte2

Procedure *SPIoutByte2 (b : byte);*
Schreibt ein Byte in den SPI-Slave.

Group: SPI SoftDriver 38

3.772 SPIoutLong

Procedure *SPIoutLong (L : longword);*

XMega:
Procedure *SPIoutLongC (L : longword);*
Procedure *SPIoutLongD (L : longword);*
Procedure *SPIoutLongE (L : longword);*
Procedure *SPIoutLongF (L : longword);*

Schreibt ein LongWord in den SPI-Slave.

Group: SPI HardwareDriver 37

3.773 SPIoutLong1

Procedure *SPIoutLong1 (L : longword);*
Schreibt ein LongWord in den SPI-Slave.

Group: SPI SoftDriver 38

3.774 SPIoutLong2

Procedure *SPIoutLong2 (L : longword);*
Schreibt ein LongWord in den SPI-Slave.

Group: SPI SoftDriver 38

3.775 SPIoutLong64

Function *SPIoutLong64C : longword;*
Function *SPIoutLong64D : longword;*
Function *SPIoutLong64E : longword;*
Function *SPIoutLong64F : longword;*

nur für **XMega**
Schreibt ein 64bit LongWord in einen SPI-Slave.

Group: SPI HardwareDriver 37

3.776 SPIoutWord

Procedure *SPIoutWord (w : word);*

XMega
Procedure *SPIoutWordC (w : word);*
Procedure *SPIoutWordD (w : word);*
Procedure *SPIoutWordE (w : word);*
Procedure *SPIoutWordF (w : word);*

Schreibt ein Word in den SPI-Slave.

Group: SPI HardwareDriver 37

3.777 SPIoutWord1

Procedure *SPIoutWord1 (w : word);*
Schreibt ein Word in den SPI-Slave.

Group: SPI SoftDriver 38

3.778 SPIoutWord2

Procedure *SPIoutWord2 (w : word);*
Schreibt ein Word in den SPI-Slave.

Group: SPI SoftDriver 38

3.779 SPIrxClear

Function *SPIrxClear : boolean;*
Die Lese und Schreib Pointer werden zurückgesetzt.

Group: SPI Network 38

3.780 SPIrxFrame

Function *SPIrxFrame : boolean;*
Empfängt einen Frame/Packet.

Group: SPI Network 38

3.781 SPIrxStat

Function *SPIrxStat : boolean;*
Gibt ein true wenn ein Frame empfangen wurde.

Group: SPI Network 38

3.782 SPItxClear

Procedure *SPItxClear;*
Lese und Schreib Pointer des TxBuffers werden zurückgesetzt.

Group: SPI Network 38

3.783 SPItxFrame

Function *SPItxFrame (**const** len : byte) : boolean;*
Sendet einen Frame/Packet.

Group: SPI Network 38

3.784 SPItxStat

Function *SPItxStat : boolean;*
Prüft ob ein zu sendender Frame gesendet ist.

Group: SPI Network 38

3.785 Sqr

Function *Sqr (f : float) : float;*
Function *Sqr (f : fix64) : fix64;*
Liefert das Quadrat des jeweiligen Arguments.

Group: Maths 5

3.786 Sqrt

Function *Sqrt (f : float) : float;*
Function *Sqrt (i : LongInt) : LongInt;*
Function *Sqrt (i : Int64) : Int64;*
Function *Sqrt (f : float) : float;*
Function *Sqrt (f : fix64) : fix64;*
Liefert die Quadratwurzel des Arguments.

Bitte beachten:
Sqrt (Fix64) braucht 600usec @16MHz,, Genauigkeit: 5 Nachkomma Stellen
Fix64Sqrt braucht 2.2msec @16MHz, Genauigkeit: 9 Nachkomma Stellen
(nur in der AVRco Profi Version verfügbar)

Group: Maths 5

3.787 SquareDivByte

Function *SquareDivByte (val, divfact : byte) : byte;*
Bildet das Quadrat eines Wertes und teilt dies durch einen zweiten Wert.

Group: Maths 5

3.788 SquareDivInt

Function *SquareDivInt (val, divfact : word|integer) : word|integer;*
Bildet das Quadrat eines Wertes und teilt dies durch einen zweiten Wert.

Group: Maths 5

3.789 SquareDivInt8

Function *SquareDivInt8 (val, divfact : int8) : int8;*
Bildet das Quadrat eines Wertes und teilt dies durch einen zweiten Wert.

Group: Maths 5

3.790 Start_Processes

Procedure *Start_Processes;*
Startet Interrupts, Prozesse und Tasks.

Xmega
erweitert mit einem Parameter, der die enabled/disabled Interrupt Levels definiert..
Procedure *Start_Processes (level : byte);*

Group: System 7

3.791 StepDestCCW

Procedure *StepDestCCW;*
Mit Rampe eine Anzahl Schritte ausführen.

Group: Stepper Driver 39

3.792 StepDestCW

Procedure *StepDestCW;*
Mit Rampe eine Anzahl Schritte ausführen.

Group: Stepper Driver 39

3.793 StepOneCCW

Procedure *StepOneCCW;*
Ein Schritt gegen den Uhrzeigersinn.

Group: Stepper Driver 39

3.794 StepOneCW

Procedure *StepOneCW;*
Ein Schritt im Uhrzeigersinn.

Group: Stepper Driver 39

3.795 StepPanicStop

Procedure *StepPanicStop;*
Während einer Rampen oder Zielfahrt sofort anzuhalten.

Group: Stepper Driver 39

3.796 StepperOff

Procedure *StepperOff;*
Schaltet die Endstufe inaktiv bzw. stromlos.

Group: Stepper Driver 39

3.797 StepperOn

Procedure *StepperOn;*
Schaltet die Endstufe aktiv.

Group: Stepper Driver 39

3.798 StepRampCCW

Procedure *StepRampCCW;*
Rampe hochlaufen gegen den Uhrzeigersinn.

Group: Stepper Driver 39

3.799 StepRampCW

Procedure *StepRampCW;*
Rampe hochlaufen im Uhrzeigersinn.

Group: Stepper Driver 39

3.800 StepRampStop

Procedure *StepRampStop;*
Abbruch eines Rampenkommandos.

Group: Stepper Driver 39

3.801 StepVelocity

Function *StepVelocity (v : word) : boolean;*
Bestimmt die aktuelle StepEnd Freuqenz.

Group: Stepper Driver 39

3.802 StrClean

Function *StrClean (**const** st : string; gt127 : boolean; subst : char) : string;*
Entfernt/ersetzt Steuerzeichen in einem String.

Group: Strings 6

3.803 StrReplace

Procedure *StrReplace (src : string; **var** dest : string; pos : byte);*
Der String *src* überschreibt den Zielstring *dest* ab der Position *pos*.

Group: Strings 6

3.804 StrToArr

Function *StrToArr (**var** st : string) : **array of** char*
Zur Generierung von Null-terminierte Strings (C Strings).

Group: Strings 6

3.805 StrToFix64

Function *StrToFix64 (st : string) : Fix64;*

konvertiert einen Strings in ein Fix64.
Verarbeitet Variablen aus dem RAM oder EEProm. Strings aus dem Flash werden nicht akzeptiert.

Groups: Strings 6, Fix64 4

3.806 StrToFloat

Function *StrToFloat (st : string) : float;*
Konvertierung eines Strings in einen Float Wert.

Group: Strings 6

3.807 StrToInt

Function *StrToInt (st : string) : byte|...|int64|word64;*
Konvertiert einen Strings in ein Int8, Byte, Word, Integer etc.

Group: Strings 6

3.808 StrToIP

Procedure *StrToIP (IPstr : String[15]; **var** Result : TIPAddr);*
Konvertiert einen IP-Adress String "aaa:bbb:ccc:ddd" in ein Byte Array.

Group: Strings 6 TINA TCP/IP 41 WizNet TCP/IP 45

3.809 StrToMAC

Procedure *StrToMAC (MACstr : String[17]; **var** Result : TMACAddr);*
Konvertiert einen MAC-Adress String "aa:bb:cc:dd:ee:ff" in ein Byte Array.

Group: Strings 6 TINA TCP/IP 41 WizNet TCP/IP 45

3.810 Succ

Function *Succ (x : type) : type;*
Gibt den nächst grösseren Wert einer Variablen zurück.

Group: Maths 5

3.811 Suspend

Procedure *Suspend (p : process|task);*
Der Prozess/Task wird angehalten.

Group: MultiTasking 5

3.812 SuspendAll

Procedure *SuspendAll (Processes, Tasks);*
Procedure *SuspendAll (Processes);*
Procedure *SuspendAll (Tasks);*
Alle angegebenen Prozesse und/oder Tasks werden gestoppt.

Group: MultiTasking 5

3.813 Swap

Function *Swap (x : type) : type;*
Vertauscht das LoByte mit dem HiByte.

Group: Maths 5

3.814 SwapIPaddr

Procedure *SwapIPaddr (**var** ip : TIPAddr);*
Spiegelt eine IP-Adresse. A3-A2-A1-A0 wird zu A0-A1-A2-A3 konvertiert.

Group: Diverse 4 TINA TCP/IP 41 WizNet TCP/IP 45

3.815 SwapLong

Function *SwapLong (**const** x : LongWord|LongInt) : LongWord|LongInt;*
Das erste und das letzte Byte und die zwei mittleren Bytes werden vertauscht.

Group: Maths 5

3.816 SwapMACaddr

Procedure *SwapMACaddr (**var** mac : TMACAddr);*
Spiegelt eine MAC-Adresse. A5-A4-A3-A2-A1-A0 wird zu A0-A1-A2-A3-A4-A5 konvertiert.

Group: Diverse 4 TINA TCP/IP 41 WizNet TCP/IP 45

3.817 SwitchKeyRepeat

Procedure *SwitchKeyRepeat1 (rept : boolean);*
Procedure *SwitchKeyRepeat2 (rept : boolean);*
Procedure *SwitchKeyRepeatG (rept : boolean);*
Die Repeat Funktion kann zur Laufzeit ein bzw. ausgeschaltet werden.

Group: SwitchPorts 39

3.818 SwitchPort_Clear

Procedure *SwitchPort1_Clear;*
Procedure *SwitchPort2_Clear;*
Procedure *SwitchPortG_Clear;*
Setzt das Port zurück.

Group: SwitchPorts 39

3.819 SysLEDallOff

Procedure *SysLEDallOff;*
Schaltet alle LED dauerhaft aus.

Group: SysLeds 40

3.820 SysLEDallOn

Procedure *SysLEDallOn;*
Schaltet alle LED dauerhaft ein.

Group: SysLeds 40

3.821 SysLEDenable

Procedure *SysLEDenable (ena : boolean);*
Schaltet den Treiber ein/aus.

Group: SysLeds

3.822 SysLEDflashAllOff

Procedure *SysLEDflashAllOff;*
Schaltet das Blinken aller LED aus.

Group: SysLeds

3.823 SysLEDflashAllOn

Procedure *SysLEDflashAllOn;*
Lässt alle LED blinken.

Group: SysLeds

3.824 SysLEDflashMsg

Procedure *SysLEDflashMsg (led, msg, rept : byte);*
Das Blinken einer Fehlermeldung.

Group: SysLeds

3.825 SysLEDflashOff

Procedure *SysLEDflashOff (b : byte);*
Schaltet das Blinken einer LED aus.

Group: SysLeds

3.826 SysLEDflashOn

Procedure *SysLEDflashOn (b : byte);*
Lässt eine LED blinken.

Group: SysLeds

3.827 SysLEDflashOnce

Procedure *SysLEDflashOnce (b : byte);*
Lässt eine LED einmal blinken.

Group: SysLeds

3.828 SysLEDflashOnOff

Procedure *SysLEDflashOnOff (b : byte; on : boolean); // b = 0..7 LED number*
Schaltet das Blinken einer LED ein oder aus.

Group: SysLeds 40

3.829 SysLEDoff

Procedure *SysLEDoff (b : byte);*
Schaltet eine LED dauerhaft aus.

Group: SysLeds 40

3.830 SysLEDon

Procedure *SysLEDon (b : byte);*
Schaltet eine LED dauerhaft ein.

Group: SysLeds 40

3.831 SysLEDonOff

Procedure *SysLEDonOff (b : byte; on : boolean); // b= 0..7 LED number*
Schaltet eine LED dauerhaft ein oder aus.

Group: SysLeds 40

3.832 System_Init

Procedure *System_Init;*
CallBack. Wird unmittelbar nach der Stack-Initialisierung ausgeführt.

Group: System 7

3.833 System_MCUCR_Init

Procedure *System_MCUCR_Init;*
CallBack. Wird unmittelbar nach dem PowerOn ausgeführt.

Group: System 7

3.834 System_Reset

Procedure *System_Reset;*
Setzt das ganze System zurück und startet es neu.

Group: System 7

3.835 System_ShutDown

Procedure *System_ShutDown;*
Fährt das System herunter

Group: System 7

3.836 SysTickDisable

Procedure *SysTickDisable;*
nur Timer Interrupt disablen, Timer läuft noch.

Group: System 7

3.837 SysTickEnable

Procedure *SysTickEnable;*
nur Timer Enablen.

Group: System 7

3.838 SysTickRestart

Procedure *SysTickRestart;*
Initialisiert den Hardware Timer komplett neu.

Group: System 7

3.839 SysTickStop

Procedure *SysTickStop;*
Disable Timer Interrupt, STOP Timer.

Group: System 7

3.840 Tan

Function *Tan (w : float) : float;*
Liefert den Tangens vom Winkel w (BogenMass) zurück.

Group: Maths 5

3.841 TanD

Function *TanD (w : float) : float;*
Liefert den Tangens vom Winkel w (GradMass) zurück.

Group: Maths 5

3.842 TestDeviceLock

Function *TestDeviceLock (d : DeviceLock) : boolean;*
Überprüft eine Semaphore, ohne diese zu verändern.

Group: Diverse 4

3.843 TickTimer2OutpEnable

Procedure *TickTimer2OutpEnable (enable : boolean; Level : byte);*
Sperrt oder gibt den Output Pin frei.
Nur für xMega vorhanden!

Group: TickTimer 41

3.844 TickTimer2RawVal

Procedure *TickTimer2RawVal (presc : byte; cmp : word|byte);*
Direkte Vorgabe von Prescaler und Compare Wert.
Nur für xMega vorhanden!

Group: TickTimer 41

3.845 TickTimer2Reload

Function *TickTimer2Reload (time : longword) : boolean;*
Verändert das Timing ohne den Timer zu stoppen.
Nur für xMega vorhanden!

Group: TickTimer 41

3.846 TickTimer2Start

Procedure *TickTimer2Start;*
Startet den Timer und seinen Interrupt, wenn vorhanden.
Nur für xMega vorhanden!

Group: TickTimer 41

3.847 TickTimer2Stop

Procedure *TickTimer2Stop;*
Stoppt den Timer und seinen Interrupt, wenn vorhanden.
Nur für xMega vorhanden!

Group: TickTimer 41

3.848 TickTimer2Time

Function *TickTimer2Time (time : longword) : boolean;*
Stoppt den Timer und setzt das Timing neu.
Nur für xMega vorhanden!

Group: TickTimer 41

3.849 TickTimerOutpEnable

Procedure *TickTimerOutpEnable (enable : boolean; Level : byte);*
Sperrt oder gibt den Output Pin frei.

Group: TickTimer 41

3.850 TickTimerRawVal

Procedure *TickTimerRawVal (presc : byte; cmp : word|byte);*
Direkte Vorgabe von Prescaler und Compare Wert.

Group: TickTimer 41

3.851 TickTimerReload

Function *TickTimerReload (time : longword) : boolean;*
Verändert das Timing ohne den Timer zu stoppen.

Group: TickTimer 41

3.852 TickTimerStart

Procedure *TickTimerStart;*
Startet den Timer und seinen Interrupt, wenn vorhanden.

Group: TickTimer 41

3.853 TickTimerStop

Procedure *TickTimerStop;*
Stoppt den Timer und seinen Interrupt, wenn vorhanden.

Group: TickTimer 41

3.854 TickTimerTime

Function *TickTimerTime (time : longword) : boolean;*
Stoppt den Timer und setzt das Timing neu.

Group: TickTimer 41

3.855 TINA_Init

Function *TINA_Init : Boolean;*
Initialisiert den Treiber.

Group: TINA TCP/IP 41

3.856 TINA_Ping

Function *TINA_Ping (PingAdr : TIPAddr; TimeOut : Word) : Word;*
Schickt ein PING.

Group: TINA TCP/IP 41

3.857 TINA_Start

Procedure *TINA_Start;*
Startet die Kommunikation.

Group: TINA TCP/IP 41

3.858 TINA_Stop

Procedure *TINA_Stop;*
Stoppt die Kommunikation.

Group: TINA TCP/IP 41

3.859 TINACreateSocket

Function *TINAcreateSocket : tSocketHandle;*
Aquiriert einen Socket.

Group: TINA TCP/IP 41

3.860 TINAfreeSocket

Procedure *TINAfreeSocket (SocketPtr : tSocketHandle);*
Gibt ein Socket Handle wieder frei.

Group: TINA TCP/IP 41

3.861 TINAinitSocket

Function *TINAinitSocket (SocketPtr : tSocketHandle) : Boolean;*
Initialisiert einen Socket.

Group: TINA TCP/IP 41

3.862 TINAlinkStat

Function *TINAlinkStat : Boolean;*
Liefert den Link-Status des ENC zurück.

Group: TINA TCP/IP 41

3.863 TINApacketReceived

Function *TINApacketReceived (SocketPtr : tSocketHandle) : Boolean;*
Fragt nach einem neuen Packet.

Group: TINA TCP/IP 41

3.864 TINAresumeReceive

Function *TINAresumeReceive (SocketPtr : tSocketHandle) : Boolean;*
Gibt den Empfang wieder frei.

Group: TINA TCP/IP 41

3.865 TINArxStat

Function *TINArxStat* (SocketPtr : tSocketHandle) : Boolean;
liefert den Receive Status eines Sockets zurück, ohne Änderungen
am Socketstatus durchzuführen

Group: TINA TCP/IP 41

3.866 TINAsendPacket

Function *TINAsendPacket (SocketPtr : tSocketHandle; Buffer : Pointer; Len : Word) : Boolean;*
Verschickt ein Packet.

Group: TINA TCP/IP 41

3.867 TINAsetPriority

Procedure *TINAsetPriority (prio : TTinaPriority);*
Setzt die Priorität des TINA Tasks.

Group: TINA TCP/IP 41

3.868 Toggle

Procedure *Toggle (b : byte; v : byte);*
Procedure *Toggle (b : Bit);*
Schaltet ein Bit um.

Procedure *Toggle (SrcDest : BitSet; op : BitSet);*

Schaltet bits in einem Bitset um.

Group: System 7

3.869 Trap

Procedure *Trap (t : byte);*
Aufruf einer Trap Prozedur.

Group: System 7

3.870 Trim

Function *Trim (**const** st : string) : string;*
Entfernt führende und nachfolgende Leerzeichen von dem String.

Group: Strings 6

3.871 TrimLeft

Function *TrimLeft (**const** st : string) : string;*
Entfernt führende Leerzeichen von dem String.

Group: Strings 6

3.872 TrimRight

Function *TrimRight (**const** st : string) : string;*
Entfernt nachfolgende Leerzeichen von dem String.

Group: Strings 6

3.873 Trunc

Function *Trunc (f : float|fix64) : integer; {LongInt}*
Beschneidet einen Wert vom Typ Float, Fix64 auf einen Integerwert.

Group: Maths 5

3.874 TWIgetBusy

Function *TWIgetBusy : boolean;*
Gibt den Status der TWI-Schnittstelle zurück.

Group: TWI (I2C) Port 43

3.875 TWIgetCmd

Function *TWIgetCmd : byte;*
Gibt das zuletzt vom Master geschickte Kommando zurück.

Group: TWI (I2C) Port

3.876 TWIgetRdy

Function *TWIgetRdy : boolean;*
Gibt Auskunft über den Sperrzustand der TWI-Schnittstelle.

Group: TWI (I2C) Port

3.877 TWIgetRxStat

Function *TWIgetRxStat : boolean;*
Stellt fest ob ein neues Packet/Daten hereingekommen ist.

Group: TWI (I2C) Port

3.878 TWIgetTxStat

Function *TWIgetTxStat : boolean;*
Stellt fest ob ein Packet/Daten abgeholt worden ist.

Group: TWI (I2C) Port

3.879 TWIinp

Function *TWIinp (SlaveAdr : byte;* ***var*** *Data) : boolean;*

XMega
Function *TWIinpC (SlaveAdr : byte;* ***var*** *Data) : boolean;*
Function *TWIinpD (SlaveAdr : byte;* ***var*** *Data) : boolean;*
Function *TWIinpE (SlaveAdr : byte;* ***var*** *Data) : boolean;*
Function *TWIinpF (SlaveAdr : byte;* ***var*** *Data) : boolean;*

Liest mindestens ein Byte aus dem angewählten Slave.

Group: TWI (I2C) Port

3.880 TWIinpP

Function *TWIinpP (SlaveAdr : byte; dst : pointer; len : word) : boolean;*

XMega
Function *TWIinpPC (SlaveAdr : byte; dst : pointer; len : word) : boolean;*
Function *TWIinpPD (SlaveAdr : byte; dst : pointer; len : word) : boolean;*
Function *TWIinpPE (SlaveAdr : byte; dst : pointer; len : word) : boolean;*
Function *TWIinpPF (SlaveAdr : byte; dst : pointer; len : word) : boolean;*

Liest mindestens ein Byte aus dem angewählten Slave.

Group: TWI (I2C) Port 43

3.881 TWIout

Function *TWIout (const SlaveAdr : byte, Command : byte|word [; Data]) : boolean;*

XMega
Function *TWIoutC (SlaveAdr : byte; Command : byte|word [; Data]) : boolean;*
Function *TWIoutD (SlaveAdr : byte; Command : byte|word [; Data]) : boolean;*
Function *TWIoutE (SlaveAdr : byte; Command : byte|word [; Data]) : boolean;*
Function *TWIoutF (SlaveAdr : byte; Command : byte|word [; Data]) : boolean;*

Schreibt mindestens ein Byte in den angewählten Slave.

Group: TWI (I2C) Port 43

3.882 TWIoutP

Function *TWIoutP (SlaveAdr : byte; Command : byte; src : pointer; len : word) : boolean;*

XMega
Function *TWIoutPC (SlaveAdr : byte; Command : byte; src : pointer; len : word) : boolean;TN* = C, D, E, F
Function *TWIoutPD (SlaveAdr : byte; Command : byte; src : pointer; len : word) : boolean;TN* = C, D, E, F
Function *TWIoutPE (SlaveAdr : byte; Command : byte; src : pointer; len : word) : boolean;TN* = C, D, E, F
Function *TWIoutPF (SlaveAdr : byte; Command : byte; src : pointer; len : word) : boolean;TN* = C, D, E, F

Schreibt mindestens ein Byte in den angewählten Slave.

Group: TWI (I2C) Port 43

3.883 TWIoutWP

Function *TWIoutWP (SlaveAdr : byte; Command : word; src : pointer; len : word) : boolean;*

XMega
Function *TWIoutWPC (SlaveAdr : byte; Command : byte; src : pointer; len : word) : boolean;*
Function *TWIoutWPD (SlaveAdr : byte; Command : byte; src : pointer; len : word) : boolean;*
Function *TWIoutWPE (SlaveAdr : byte; Command : byte; src : pointer; len : word) : boolean;*
Function *TWIoutWPF (SlaveAdr : byte; Command : byte; src : pointer; len : word) : boolean;*

Schreibt mindestens ein Byte in den angewählten Slave.

Group: TWI (I2C) Port 43

3.884 TWIrxClear

Procedure *TWIrxClear;*
Das Statusbyte des RxBuffers wird zurückgesetzt.

Group: TWI Network 44

3.885 TWIrxFrame

Function *TWIrxFrame (node : byte) : boolean;*
Liest einen Frame von einem Slave aus.

Group: TWI Network 44

3.886 TWIrxStat

Function *TWIrxStat : boolean;*
Gibt ein true zurück, wenn ein Frame empfangen wurde.

Group: TWI Network 44

3.887 TWIsetBusy

Function *TWIsetBusy (busy : boolean) : boolean;*
Sperrt oder gibt TWI-Schnittstelle wieder frei.

Group: TWI (I2C) Port 43

3.888 TWIsetGC

Procedure *TWIsetGC (BroadcastEnable : boolean);*
Bestimmt ob der Slave Broadcast Telegramme empfangen soll.

Group: TWI (I2C) Port 43

3.889 TWIsetRdy

Function *TWIsetRdy (ready : boolean) : boolean;*
Setzt den Status des RxBuffers auf leer und den TxBuffer auf voll.

Group: TWI (I2C) Port 43

3.890 TWIsetSlaveAddr

Function *TWIsetSlaveAddr (SlaveAddr : byte);*
Legt eine neue Slave Adresse fest.

Group: TWI (I2C) Port 43

3.891 TWIstat

Function *TWIstat (SlaveAdr : byte) : boolean;*

XMega
Function *TWIstatC (SlaveAdr : byte) : boolean;*
Function *TWIstatD (SlaveAdr : byte) : boolean;*
Function *TWIstatE (SlaveAdr : byte) : boolean;*
Function *TWIstatF (SlaveAdr : byte) : boolean;*

Stellt fest, ob der gewählte Slave vorhanden ist.

Group: TWI (I2C) Port 43

3.892 TWItxBroadcast

Function *TWItxBroadcast (cmd : byte; subnode : byte) : boolean;*
Sendet einen I2C Broadcast an alle Slaves.

Group: TWI Network 44

3.893 TWItxClear

Procedure *TWItxClear;*
Das Statusbyte des TxBuffers wird zurückgesetzt.

Group: TWI Network 44

3.894 TWItxFrame

Function *TWItxFrame (node : byte; len : byte[word]) : boolean;*
Sendet ein Packet.

Group: TWI Network 44

3.895 TWItxStat

Function *TWItxStat : boolean;*
Prüft, ob ein zu sendender Frame gesendet ist.

Group: TWI Network 44

3.896 uDelay

Procedure *uDelay (d : byte);*
Software Delay in usec x 10.

Group: System 7

3.897 uDelay_1

Procedure *uDelay_1 (d : byte);*
Software Delay in 1 usec.

Group: System 7

3.898 UnLock

Procedure *UnLock (p : process);*
Ein gelockter Prozess gibt den Prozessor wieder frei.

Group: MultiTasking 5

3.899 UpCase

Function *Upcase (ch : char) : char;*
Wandelt char in Grossbuchstaben.

Group: Strings 6

3.900 UpperCase

Function *Uppercase (st : string) : string;*
Wandelt *String* in Grossbuchstaben.

Group: Strings 6

3.901 UsrDevPtr

Function *UsrDevPtr (p:pointer): pointer;*
Der Pointer soll ins User Device zeigen.

Group: System 7

3.902 ValueInRange

Function *ValueInRange (value, vmin, vmax : type) : boolean;*
Vergleicht die Variable "value" mit den beiden Grenzen "vmin" und "vmax".
Type = Enum, Byte, Int8, Char, Word, Integer, Longword, Longint, Word64, Int64, Fix64,Float

Group: Maths 5

3.903 ValueInTolerance

Function *ValueInTolerance (value, ref, tol : type) : boolean;*
Vergleicht die Variable "value" mit den Grenzen.

Group: Maths 5

3.904 ValueInToleranceP

Function *ValueInToleranceP (value, ref : type; tol : byte) : boolean;*
Vergleicht die Variable "value" mit den Grenzen.

Group: Maths 5

3.905 ValueTrimLimit

Function *ValueTrimLimit (value, vmin, vmax : type) : type;*
Gibt das Ergebnis zurück das in den Grenzen *vmin/vmax* liegt.
Type = Enum, Byte, Int8, Char, Word, Integer, Longword, Longint, Word64, Int64, Fix64, Float

Group: Maths 5

3.906 WaitDeviceFree

Function *WaitDeviceFree (s : DeviceLock [; timeout: word]) : boolean;*
Ein Prozess/Task legt sich schlafen bis das Device frei ist.

Group: MultiTasking 5

3.907 WaitPipe

Function *WaitPipe (p : pipe [; timeout: word]) : boolean;*
Ein Prozess/Task schaltet sich inaktiv, bis eine spezielle Pipe Daten hat.

Group: MultiTasking 5 Pipes 27

3.908 WaitSema

Function *WaitSema (s : semaphore [; timeout: word]) : boolean;*
Ein Prozess/Task schaltet sich inaktiv, bis eine spezielle Semaphore > 0 ist.

Group: MultiTasking 5

3.909 WatchDogStart

Procedure *WatchDogStart;*
Initialisiert und startet den WatchDog.

Group: Diverse 4

3.910 WatchDogStop

Procedure *WatchDogStop;*
Stoppt den WatchDog.

Group: Diverse 4

3.911 WatchDogTrig

Procedure *WatchDogTrig;*
Triggert den Hardware WatchDog der CPU.

Group: Diverse 4

3.912 Within

Function *Within (lo, x, hi : type) : type;*
Prüft eine Zahl gegen zwei Grenzwerte ab. Alle Typen Argumente müssen identisch sein.
type: Byte, Int8, Word, Integer, Longint, Longword, Int64, Word64, Fix64, Float.

Group: Maths 5

3.913 WordToBCD

Function *WordToBCD (w : word) : word;*
Konvertiert ein Word in das BCD Format.

Group: Maths 5

3.914 Write

Procedure *Write (DeviceFunc : function; arg : char|string);*
Stringausgabe über eine Prozedur.

Group: Diverse 4

3.915 WriteLn

Procedure *WriteLn(DeviceFunc : function; arg : char|string);*
Stringausgabe über eine Prozedur. CRLF wird angehängt.

Group: Diverse 4

3.916 wzAcceptConnection

Function *wzAcceptConnection (SocketPtr : tSocketHandle; YesNo : boolean) : boolean;*
Hat sich ein Client connected, kann man diese Verbindung akzeptieren oder abbrechen.

Group: WizNet TCP/IP 45

3.917 wzClientConnected

Function *wzClientConnected (SocketPtr : tSocketHandle) : boolean;*
Ein ***Server*** pollt solange diese Funktion bis ein true zurück kommt.

Group: WizNet TCP/IP 45

3.918 wzConnect

Function *wzConnect (SocketPtr : tSocketHandle) : boolean;*
Baut die Verbindung zu einem Server auf.

Group: WizNet TCP/IP 45

3.919 wzCreateSocket

Function *wzCreateSocket : tSocketHandle;*
Fordert ein Socket Handle an.

Group: WizNet TCP/IP 45

3.920 wzDisConnect

Function *wzDisConnect (SocketPtr : tSocketHandle) : boolean;*
Löst eine Verbindung zwischen Server und Client.

Group: WizNet TCP/IP 45

3.921 wzDNSqueryHost

Function *wzDNSqueryHost (Buffer : pointer; BuffLen : word; Hostname : pointer to string;*
var *Result_IP : tIPaddr) : boolean;*
Erstellt den DNS Clienten und erfragt eine IP Adresse.

Group: WizNet TCP/IP 45

3.922 wzFreeSocket

Procedure *wzFreeSocket (SocketPtr : tSocketHandle);*
Freigabe eines Socket Handles.

Group: WizNet TCP/IP 45

3.923 wzGetLastError

Function *wzGetLastError (SocketPtr : tSocketHandle) : TwzStatus;*
Gibt den Fehlerstatus der letzten Operation zurück.

Group: WizNet TCP/IP 45

3.924 wzGetSocketState

Function *wzGetSocketState (SocketPtr : tSocketHandle) : TwzStatus;*
Gibt den aktuellen Status eines Socket zurück.

Group: WizNet TCP/IP 45

3.925 wzInit

Function *wzInit : boolean;*
Führt einen Software Reset aus und macht eine Grund-Initialisierung.

Group: WizNet TCP/IP 45

3.926 wzInitSocket

Function *wzInitSocket (SocketPtr : tSocketHandle) : boolean;*
Nur für **UDP**. Grund Initialisierung des Sockets im UDP Mode.

Group: WizNet TCP/IP 45

3.927 wzListen

Function *wzListen (SocketPtr : tSocketHandle) : boolean;*
Aktiviert einen Server.

Group: WizNet TCP/IP 45

3.928 wzPacketReceived

Function *wzPacketReceived (SocketPtr : tSocketHandle) : boolean;*
Gibt ein true zurück wenn ein Packet empfangen wurde.

Group: WizNet TCP/IP 45

3.929 wzReceiveBuffer

Function *wzReceiveBuffer (SocketPtr : tSocketHandle): word;*
Holt Daten ab.

Group: WizNet TCP/IP 45

3.930 wzReInitSocket

Function *wzReInitSocket (SocketPtr : tSocketHandle) : boolean;*
Nur für **UDP**. Ähnlich wzInitSocket.

Group: WizNet TCP/IP 45

3.931 wzReset

Procedure *wzReset;*
Führt einen Software Reset auf dem W3100A Chip aus.

Group: WizNet TCP/IP 45

3.932 wzResumeReceive

Function *wzResumeReceive (SocketPtr : tSocketHandle) : boolean;*
Gibt den Empfang wieder frei.

Group: WizNet TCP/IP 45

3.933 wzSendBuffer

Function *wzSendBuffer (SocketPtr : tSocketHandle; Buffer : Pointer; Len : word): boolean;*
Sende Auftrag an den Kernel.

Group: WizNet TCP/IP 45

3.934 wzSetDNSserver

Procedure *wzSetDNSserver (IPAddr : tIPaddr);*
Bestimmt die IP-Adresse eines DNS Servers.

Group: WizNet TCP/IP 45

3.935 wzSetGatewayAddr

Procedure *wzSetGatewayAddr (IPAddr : tIPaddr);*
Bestimmt die Gateway IP-Adresse für diesen Stack.

Group: WizNet TCP/IP 45

3.936 wzSetHWAddr

Procedure *wzSetHWAddr (MacAddr : TMacAddr);*
Bestimmt die MAC Adresse dieses Stacks.

Group: WizNet TCP/IP 45

3.937 wzSetIPAddr

Procedure *wzSetIPAddr (IPAddr, Mask : tIPaddr);*
Bestimmt die lokale IP-Adresse und die Subnet Mask dieses Stacks.

Group: WizNet TCP/IP 45

3.938 wzSetPriority

Procedure *wzSetPriority (prio : TwzPriority);*
Bestimmt die Priorität des internen wzTasks.

Group: WizNet TCP/IP 45

3.939 wzSetRetryCount

Procedure *wzSetRetryCount (Retry : byte);*
Bestimmt die Anzahl der Retries bei Fehlern.

Group: WizNet TCP/IP 45

3.940 wzSetSNTPserver

Procedure *wzSetSNTPserver (IPAddr : tIPaddr);*
Bestimmt die IP-Adresse eines SNTP Servers.

Group: WizNet TCP/IP 45

3.941 wzSetTimeOut

Procedure *wzSetTimeOut (RetryTimeout : word);*
Bestimmt die Pause zwischen den Retries.

Group: WizNet TCP/IP 45

3.942 wzSNTPqueryDateTime

Function *wzSNTPqueryDateTime (Buffer : pointer; BuffLen : word;*
var *DateTime : tDateTime) : boolean;*
Erstellt den SNTP Clienten und fragt Datum und Uhrzeit ab.

Group: WizNet TCP/IP 45

3.943 wzTelnetClose

Procedure *wzTelnetClose;*
Schaltet den Server ab. Einschalten mit wzTelnetListen.

Group: WizNet TCP/IP 45

3.944 wzTelnetConnected

Function *wzTelnetConnected : boolean;*
Gibt den connect Status des Severs zurück.

Group: WizNet TCP/IP 45

3.945 wzTelnetCreate

Function *wzTelnetCreate : boolean;*
Erstellt den Telnet Server.

Group: WizNet TCP/IP 45

3.946 wzTelnetEcho

Procedure *wzTelnetEcho (EchoOn : boolean);*
Schatet das Echo des Telnet Servers ein/aus.

Group: WizNet TCP/IP

3.947 wzTelnetFree

Procedure *wzTelnetFree;*
Gibt den Telnet Socket wieder frei.

Group: WizNet TCP/IP

3.948 wzTelnetGetClient

Function *wzTelnetGetClient (**var** ClientIP : tIPaddr; **var** ClientPort : word) : boolean;*
Abfrage der IP-Adresse und Port eines Clienten.

Group: WizNet TCP/IP

3.949 wzTelnetGetState

Function *wzTelnetGetState : TwzStatus;*
Status Abfrage Funktion ohne Auswirkungen auf den Status des Servers.

Group: WizNet TCP/IP

3.950 wzTelnetIdleTimeout

Procedure *wzTelnetIdleTimeout (tnTimeout : byte);*
Setzt den Timeout für einen inaktiven Clienten.

Group: WizNet TCP/IP

3.951 wzTelnetListen

Function *wzTelnetListen : boolean;*
Schaltet nach dem Create den Server in Listen Mode.

Group: WizNet TCP/IP

3.952 wzTelnetRead

Function *wzTelnetRead (Prompt : tTelnetStr) : tTelnetStr;*
Der Telnet Servers holt ein Kommando vom Client.

Group: WizNet TCP/IP

3.953 wzTelnetWrite

Function *wzTelnetWrite (Str : tTelnetStr) : boolean;*
Sende Auftrag an den Server.

Group: WizNet TCP/IP 45

3.954 wzTelnetWriteLn

Function *wzTelnetWriteLn (Str : tTelnetStr) : boolean;*
Identisch mit wzTelnetWrite, hängt ein CRLF an den String an.

Group: WizNet TCP/IP 45

4 Reservierte Schlüsselwörter

4.1 absolute

absolute
Variable Overlay.

```
var
  abc  : byte;
  ovr1 : byte absolute abc;                  // var abc overlayed
```

Group: Reserved Words 6

4.2 and

and

Operator. AND Maske.
*a := a **and** $0f;*

Group: Reserved Words 6

4.3 array

array

Strukturierter Typ aus einer festen Anzahl von Komponenten vom gleichen Typ. 1..4 Dimensionen. Eine Dimension kann bis zu 61440 ($F000) Mitglieder haben. Totale Grösse beschränkt auf ca. 60kBytes.
Typen: Bytes, Chars, Int8, Booleans, Words, Integers, LongWords, Floats, Fix64, Pointers, Procedures.

```
type Day = (Mon,Tue,Wed,Thu,Fri,Sat,Sun);

var
  WorkHour   : array[1..8] of Integer;
```

```
Week      : array [ 1.. 7] of Day;
ArrEn     : array [ Day] of byte;
```

Group: Reserved Words 6

4.4 asm

asm

Assembler Statement(s).

```
ASM:  PUSH _ACCA;
```

```
ASM;
  LDI _ACCA, 040h
  ANDI _ACCA, myProg.a;                  //a = Pascal var in myProg
ENDASM;
```

Group: Reserved Words 6

4.5 assign

assign

reserviert für geplante Erweiterungen.

Group: Reserved Words 6

4.6 at

at

reserviert für geplante Erweiterungen.

Group: Reserved Words 6

4.7 begin

begin

Start Prozedur-, Funktions-, Process- oder Task Rumpf.

Group: Reserved Words 6

4.8 bitset

bitset

8bit oder 16bit, abhängig von der zugrunde liegenden Enumeration.

```
type
        TbitNames       = (one, two, three, four, five, six);          // enum
```

```
        TBitSet = BitSet of TbitNames;          // build a bitset type
var
        BitSet1 : TBitSet;                      // build a bitset var
        BitSet2 : TBitSet;                      // build a bitset var
```

Group: Reserved Words 6

4.9 boolean

boolean

Vordefinierter Typ. true = $FF, false = $00
var flag : boolean;

Group: Reserved Words 6

4.10 break

break

Abbruch einer Eine FOR-, WHILE-, REPEAT-Schleife. Siehe auch for repeat while

```
for x := 1 to 9  do
  ...
  if a:= 0 then Break;
  ...
endfor;
```

Group: Reserved Words 6

4.11 by

by

optional bei FOR-Schleifen. Bestimmt den Inkrement- bzw. Dekrement-Wert (1..255).

```
for x :=1 to 15 by 2  do
for x :=16 downto 0 by 2  do
```

Group: Reserved Words 6

4.12 byte

byte

1. Vordefinierter Type
Ganze Zahlen. 0 bis 255. Benötigen ein Byte im Speicher.

2. Typecast
Wandelt das Argument in ein Byte.

```
Function Byte (a : type) : byte;
  b:= byte (x);
```

Group: Reserved Words 6

4.13 case

case

einleitendes Statement für einen Verzweigungsblock.

```
Case x of
0       : inc(a);
          |
1, 7    : a:= 4;
          |
2..6    : x:= x + a;
           dec(x);
          |
else      x:= 0;
EndCase;
```

Group: Reserved Words 6

4.14 char

char

1. Vordefinierter Type
Zeichen des ASCII Zeichensatz. Benötigt ein Byte im Speicher.

```
const  cd      : char   = 'D';
       Bell    : char   = ^G;          {Control G}
       LF      : char   = #10;         {Line Feed}
```

2. Typecast
Wandelt das Argument in ein Char.

```
Function Char (a : type) : char;
  ch:= char (x);
```

Group: Reserved Words 6

4.15 chr

chr

reserviert für geplante Erweiterungen.

Group: Reserved Words 6

4.16 class

class

reserviert für geplante Erweiterungen.

Group: Reserved Words

4.17 close

close

reserviert für geplante Erweiterungen.

Group: Reserved Words

4.18 const

const

Beginn einer Konstanten Deklaration.
```
const abc : byte = 1;
```

Group: Reserved Words

4.19 continue

continue

Setzt die Programmausführung mit dem nächsten Durchlauf einer **for**-, **while**- oder **repeat**-Schleife fort.
Die nachfolgenden Statements bis zu EndFor, EndWhile oder Until werden ignoriert.
Siehe auch for repeat while

```
for x:= 1 to 9 by 2 do
  ...
  if a:= 0 then Continue;
  ...
endfor;
```

Group: Reserved Words

4.20 define

define

Bestimmte Imports benötigen eine zusätzliche Spezifikation, die mittels *Define* durchgeführt wird.
```
Define ProcClock = 4000000;              //4 Mhz
```

Group: Reserved Words 6

4.21 define_fuses

define_fuses

Optional können damit die Fuses für die Programmer in der Source vorgegeben werden.

```
Define_Fuses
Override_Fuses;        // optional, always replaces fuses in ISPE
COMport = COM1;        // COM2..COM7, USB
Supply = 4.0, 200;     // programmer supplies target, 4.0Volt, 200mA
SPIclk = 1000000;      // optional SPI programming speed
NoteBook = B;          // A ... N
LockBits0 = [];
LockBits1 = [];
FuseBits0 = [CKSEL1];
FuseBits1 = [BOOTRST, BOOTSZ1, SPIEN, OCDEN];
FuseBits2 = [];
ProgMode = SPI;        // SPI, JTAG or OWD
ProgFuses = true;      // or false – program Fuse Bits (* REV4 *)
ProgLock = true;       // or false – program Lock Bits (* REV4 *)
ProgFlash = true;      // or false – program Flash (* REV4 *)
ProgEEprom= true;      // or false – program Eeprom (* REV4 *)
CalByte = 2, $3FFF;    // read/write calibration byte (* REV4 *)
AddApp = 'pathname\projectname';
```

Group: Reserved Words 6

4.22 define_usr

define_usr

Damit können bei Bedarf Konstante definiert werden, die auch von jeder Unit aus sichtbar und zugreifbar sind.

```
Define_USR myConst = 1;
```

Group: Reserved Words 6

4.23 definefrom

definefrom

Wenn Units importiert sind, kann eine Definition auch innerhalb einer Unit erfolgen.

```
DefineFrom unit1;                    // Definitionen von Unit1
```

Group: Reserved Words 6

4.24 device

device

Prozessor und Hardware Spezifikation.
device *= 90S2313;*

Group: Reserved Words 6

4.25 devicelock

devicelock

Vordefinierter Typ.
Für Device Treiber um innerhalb MultiTasking diese Treiber gegen andere Prozesse/Tasks zu sperren.

Group: Reserved Words 6

4.26 div

div

Operator. Divison ganzer Zahlen.
Zulässige Operanten: Typ Byte, Int8, Boolean, Integer, Word, Longint, Longword, Word64, Int64 oder Fix64.
a:= a ***div*** *b;*

Group: Reserved Words 6

4.27 do

do

leitet in for-, while-, with-Statements den Anweisungsteil ein. Siehe for while with

Group: Reserved Words 6

4.28 downto

downto

bestimmt Decrement der Laufvariable in for-Schleifen. Siehe for

Group: Reserved Words 6

4.29 els_if

els_if
elsif

Siehe elsif

Group: Reserved Words 6

4.30 else

else

Anweisung in IF Statements
if a > b ***then*** *a:= b;* ***else*** *b:= a;* ***endif****;*

Group: Reserved Words 6

4.31 elsif

elsif
els_if

Anweisung in IF Statements
if a > b ***then*** *..;* ***elsif*** *b = a* ***then*** *..;* ***endif****;*

Group: Reserved Words 6

4.32 end

end

Ende Prozedur-, Funktions-, Task- oder Prozess-Rumpf.

Group: Reserved Words 6

4.33 end_asm

end_asm
endasm

Siehe endasm

Group: Reserved Words 6

4.34 end_case

end_case
endcase

Siehe endcase

Group: Reserved Words 6

4.35 end_for

end_for
endfor

Siehe endfor

Group: Reserved Words 6

4.36 end_if

end_if
endif

Siehe endif

Group: Reserved Words 6

4.37 end_loop

end_loop
endloop

Siehe endloop

Group: Reserved Words 6

4.38 end_try

end_try
endtry

Siehe endtry

Group: Reserved Words 6

4.39 end_while

end_while
endwhile

Siehe endwhile

Group: Reserved Words 6

4.40 end_with

end_with
endwith

Siehe endwith

Group: Reserved Words 6

4.41 endasm

endasm
end_asm

Ende eines Assembler Textes. Siehe auch asm

Group: Reserved Words 6

4.42 endcase

endcase
end_case

Abschluss eines case Statements. Siehe case

Group: Reserved Words 6

4.43 endfor

endfor
end_for

Abschluss eines for Statements.Siehe for

Group: Reserved Words 6

4.44 endif

endif
end_if

Abschluss eines if Statements. Siehe if

Group: Reserved Words 6

4.45 endloop

endloop
end_loop

Abschluss eines loop Statements. Siehe loop

Group: Reserved Words 6

4.46 endtry

endtry
end_try

Abschluss eines try Statements. Siehe try

Group: Reserved Words 6

4.47 endwhile

endwhile
end_while

Abschluss eines while Statements. Siehe while

Group: Reserved Words 6

4.48 endwith

endwith
end_with

Abschluss eines with Statements. Siehe with

Group: Reserved Words 6

4.49 eof

eof

reserviert für geplante Erweiterungen.

Group: Reserved Words 6

4.50 eoln

eoln

reserviert für geplante Erweiterungen.

Group: Reserved Words 6

4.51 except

except

Option bei try Statement. Siehe try

Group: Reserved Words 6

4.52 exec

exec

reserviert für geplante Erweiterungen.

Group: Reserved Words

4.53 exit_loop

exit_loop
exitloop

Siehe exitloop

Group: Reserved Words

4.54 exitloop

exitloop
exit_loop

Verlassen einer Loop-EndLoop Schleife. Siehe loop

Group: Reserved Words

4.55 export

export

reserviert für geplante Erweiterungen.

Group: Reserved Words

4.56 false

false

Wert einer bool'schen Konstante. Vordefiniert = 0.

Group: Reserved Words

4.57 file

file

Schlüsselwort im Filesystem.
Var *ff :* ***file of*** *Byte*

Group: Reserved Words

4.58 finalization

finalization

Optionaler Teil einer Unit.
Die in diesem Block vorhandenen Statements werden abgearbeitet, wenn die Applikation die System
Prozedur "*System_ShutDown*" aufruft. Siehe System_ShutDown

Group: Reserved Words 6

4.59 float

float

1. Vordefinierter Type
32bit, 4 Bytes, 6..9 Digits, 10E-38..10E38
```
var f : float;
```

2. Typecast
Wandelt das Argument in ein Float.
```
Function Float (a : type) : float;
  f:= Float (x);
```

Group: Reserved Words 6

4.60 for

for

Einleitendes Statement für eine for Schleife.
```
for x:= a to v1 do {ramp up}
  PWMport1:= x;
endfor;

for x:= x downto 0 do {ramp down}
  PWMport1:= x;
endfor;
```

Group: Reserved Words 6

4.61 forward

forward

Vorwärtsreferenz von Pointern, Funktionen, Prozeduren, Tasks und Prozessen.
```
Procedure Test1; Forward;
```

Group: Reserved Words 6

4.62 from

from

Mit FROM wird gezielt aus einer bestimmten Bibliothek ein Typ, Function oder Procedure importiert.
From *System* ***Import*** *LongInt, LongWord, Float;*

Group: Reserved Words 6

4.63 function

function

Funktion Deklaration.
Function *Test (****var*** *par : byte) : byte;*
var *loc : boolean;* *//lokale Variable*
begin
…
end;

Group: Reserved Words 6

4.64 goto

goto

Absolute Sprunganweisung innerhalb eines Blocks.
begin
Label: *lab1;* *//Definition des Labels*
...
Goto *Lab1;*

Group: Reserved Words 6

4.65 idle

idle

reserviert für geplante Erweiterungen.

Group: Reserved Words 6

4.66 if

if

Einleitendes Statement für eine Verzweigung.
if a > b **then**
...
elsif a = b **then**
...

```
else {a ist kleiner b}
  ...
endif;
```

Group: Reserved Words 6

4.67 implementation

implementation

1. in Main
Programm Start. Der Compiler fügt hier den Reset-Code und die Initialisierung ein.

2. in Units
Dieser Bereich enthält lokale Variablen, Konstante und Type Deklarationen einer Unit, sowie deren Prozeduren und Funktionen. Siehe unit

Group: Reserved Words 6

4.68 import

import

Import von **Hardware abhängigen** Systemfunktionen. Siehe from

Group: Reserved Words 6

4.69 in

in

1. Spezifikation des Speicherorts von Units. Siehe uses

2. Auswertung von Enums und ordinalen Typen.
if v2 in ['a'..'g'] then ...
if x in [45..56] then ..

Group: Reserved Words 6

4.70 inherit

inherit

Import bzw. Vererbung eines bestehenden Records in einen neuen Record.
inherit *RecordName*;

Group: Reserved Words 6

4.71 inherited

inherited

reserviert für geplante Erweiterungen.

Group: Reserved Words

4.72 initialization

initialization

Optionaler Teil einer Unit.

Group: Reserved Words

4.73 int64

int64

Vordefinierter Type. 64bit, 8 Bytes, -9223372036854775808 ... 9223372036854775807
```
from System Import Int64;
var i64 : int64;
```

Group: Reserved Words

4.74 int8

int8
ShortInt

1. Vordefinierter Type. Short Integer.
8bit, 1 Byte, -128..127
```
var i8 : int8;
    shint : ShortInt;
```

2. Typecast
Wandelt das Argument in ein ShortInteger.
```
Function Int8 (a : type) : int8;
 i:= Int8 (x);
 i:= ShortInt (x);
```

Group: Reserved Words

4.75 integer

integer

1. Vordefinierter Type
16bit, 2 Bytes, -32768..32767
```
var i : integer;
```

2. Typecast
Wandelt das Argument in ein Integer.
```
Function Integer (a : type) : integer;
  i:= Integer (x);
```

Group: Reserved Words 6

4.76 interface

interface

Abschnitt innerhalb einer Unit. Deklariert Konstanten, Typen, Variablen, Prozeduren und Funktionen, die für Clients verfügbar sind. Siehe unit

Group: Reserved Words 6

4.77 interrupt

interrupt

Deklaration einer Interrupt Prozedur.
Die jeweiligen CPU und I/O-Steuerregister für den spezifischen Interrupt müssen von der Applikation zusäzlich (wie im AVR Controller Manual beschrieben) initialisiert werden.

```
Interrupt Int0;
begin
  IncSema (sema0);
end;
```

Group: Reserved Words 6

4.78 label

label

Sprungziel für eine goto Anweisung. Siehe goto

Group: Reserved Words 6

4.79 locked

locked

Schützt globale ordinale Variablen gegen konkurrierende Zugriffe von Interrupts, Prozesse oder Tasks.
```
Var i : integer, locked;
```

Group: Reserved Words 6

4.80 longint

longint

1. Vordefinierter Type
32bit, 4 Bytes, -2147483648..2147483647
var *li : longint;*

2. Typecast
Wandelt das Argument in ein LongInt.
Function *LongInt (a : type) : longint;*
Li:= LongInt (x);

Group: Reserved Words 6

4.81 longword

longword

Vordefinierter Type. 32bit, 4 Bytes, 0..4294967295
var *lw : longword;*

Group: Reserved Words 6

4.82 loop

loop

Beginn einer Endlosschleife.
```
begin
  loop
  ...
  if ... then exitloop endif;
  endloop;
  ...                              // Statements nach "exitloop"
end.
```

Group: Reserved Words 6

4.83 mod

mod

Operator. Modulo ganzer Zahlen.
erg:= a **mod** *5;*
wenn "a" negativ ist, wird auch "erg" negativ.
Um immer ein positives Ergebnis zu erhalten muss Abs benutzt werden.

Group: Reserved Words 6

4.84 nil

nil

Vordefinierte Konstante vom Typ Pointer = $0000.
p := nil;

Group: Reserved Words 6

4.85 not

not

Operator. Invertierung von Variablen.
Zulässige Operanten: Typ Byte, Int8, Boolean, Integer, Word, Longint, Longword, Word64, Int64 oder Fix64.
*a:= **not** a;*

Group: Reserved Words 6

4.86 object

object

reserviert für geplante Erweiterungen.

Group: Reserved Words 6

4.87 of

of

Festlegung von (ggf. vordefinierten) Typen. Siehe array

Group: Reserved Words 6

4.88 or

or

Operator. OR Maske.
*a:= a **or** $30;*

Group: Reserved Words 6

4.89 override

override

reserviert für geplante Erweiterungen.

Group: Reserved Words 6

4.90 pidcontrol

pidcontrol

Pseudorecord für PID Regler.
var *Pid1 : PIDcontrol[iLimit, dIntVal];*

Group: Reserved Words 6

4.91 pipe

pipe

Vordefinierter Typ. FIFO Speicher.
var *Pipe1 : Pipe[16]* ***of*** *byte;*

Group: Reserved Words 6

4.92 pointer

pointer

1. Vordefinierter Type
16bit, 2 Bytes
Adresse einer Variablen.

```
type    tpb : pointer to Byte;
var     pb : tpb;
...
pb:= tpb (irgendeinPointer);
```

2. Typecast
Wandelt das Argument in einen Pointer.

```
Function Pointer (a : type) : pointer;
p:= Pointer (x);
```

Group: Reserved Words 6

4.93 procedure

procedure

16bit, Parameter, word, Adresse.

```
var proc : procedure;
procedure indirtest;
begin
  ...
end;

begin {Main Program}
  ...
```

```
  Proc:= @indirtest;                {Variable besetzen mit Adresse von indirtest}
  Proc;                             {indirtest aufrufen}
  ...
end.
```

Group: Reserved Words 6

4.94 process

process

Prozess Deklaration.
Process *ProcessName (StackSize, FrameSize : word; DataPage);*

```
Process Proc1 (32, 32 : iData);                 // default prio=3, autostart
Process Proc1 (32, 32 : iData; 5);              // priority 5
Process Proc1 (32, 32 : iData; resumed);                // default prio=3, automatic start
Process Proc1 (32, 32 : iData; 5, suspended);   // prio= 5, no automatic start
```

(Der Main Process hat Default Priority = 5).

Group: Reserved Words 6

4.95 program

program

Start des Programms.
Program *Test;*

Group: Reserved Words 6

4.96 record

record

Vordefinierter Typ. Struktur mit Feldern.

```
type    tMonths = (Jan,Feb,Mar,Apr,May,Jun,July,Aug,Sep,Oct,Nov,Dec);
var
        Date = record
                 Day : byte;
                Month : tMonths
                 Year : word;
               end;
```

Group: Reserved Words 6

4.97 repeat

repeat

Einleitendes Statement für eine Programmschleife.

```
repeat
  inc(TCC);
```

```
until TCC > 20;
```

Group: Reserved Words 6

4.98 return

return

Abbruch und Exit innerhalb Prozedur/Funktion.

```
Procedure Test1;
begin
  Statement ...;
    if (a > b) then
      Return;
   endif;
  Statement ...;
end;
```

Group: Reserved Words 6

4.99 rol

rol

Operator. Links rotieren. Alle Bits bleiben erhalten, tauschen aber ihre Positionen.

```
a:= a rol 4;
```

Group: Reserved Words 6

4.100 ror

ror

Operator. Rechts rotieren. Alle Bits bleiben erhalten, tauschen aber ihre Positionen.

```
a:= a ror 4;
```

Group: Reserved Words 6

4.101 semaphore

semaphore

Vordefinierter Typ. 8bit, Byte. Wird durch Processes oder Tasks importiert.

```
var sema : Semaphore;
```

Group: Reserved Words 6

4.102 sendsema

sendsema

reserviert für geplante Erweiterungen.

Group: Reserved Words 6

4.103 set

set

reserviert für geplante Erweiterungen.

Group: Reserved Words 6

4.104 shl

shl

Operator. Links schieben. Schiebt alle Bits und füllt dabei die freiwerdenden Bits mit '0' auf.
*a:= a **shl** 5;*

Group: Reserved Words 6

4.105 shla

shla

Operator. Arithmetisch links schieben. Schiebt Bit 6..0 und füllt dabei die freiwerdenden Bits mit '0' auf.
Das höchstwertige Bit bleibt unverändert.
*a:= a **shla** 5;*

Group: Reserved Words 6

4.106 shortint

ShortInt

siehe int8

4.107 shr

shr

Operator. Rechts schieben. Schiebt alle Bits und füllt dabei die freiwerdenden Bits mit '0' auf.
*a:= a **shr** 5;*

Group: Reserved Words 6

4.108 shra

shra

Operator. Arithmetisch rechts schieben. Schiebt Bit 6..0 und füllt dabei die freiwerdenden Bits mit dem Wert des höchstwertigen Bit auf. Das höchstwertige Bit bleibt unverändert.
*a:= a **shra** 5;*

Group: Reserved Words 6

4.109 str

str

reserviert für geplante Erweiterungen.

Group: Reserved Words 6

4.110 string

string

Vordefinierter Typ. 0..255 bytes, Variable oder Konstante. Zeichenkette.

```
var     st1 : string[8];
        …
        st1:= st;
```

Group: Reserved Words 6

4.111 structconst

structconst

strukturierte Konstante.

```
structconst             //Konstante im Rom, beim Startup ins Ram kopiert
  str : st10 = 'abcde';
```

Group: Reserved Words 6

4.112 systimer

systimer

16bit, Word. Wird durch SysTick importiert. Wird bei jedem System Tick dekrementiert, falls > 0.
var Timer1 : SysTimer;

Group: Reserved Words 6

4.113 systimer32

systimer32

32bit, LongWord. Wird durch SysTick importiert. Wird bei jedem System Tick dekrementiert, falls > 0.

```
from System import longword;
var Timer3 : SysTimer32;
```

Group: Reserved Words

4.114 systimer8

systimer8

8bit, Byte. Wird durch SysTick importiert. Wird bei jedem System Tick dekrementiert, falls > 0.

```
var Timer2 : SysTimer8;
```

Group: Reserved Words

4.115 table

table

Vordefinierter Typ.1 Dimension. Bis zu 255 Mitglieder.Die Table Länge ist auf 2er Potenzen beschränkt
Typen: Bytes, Chars, Int8, Booleans, Words, Integers, LongWords, Floats, Fix64, Pointers, Procedures.

```
var     tb1 : Table[0..15] of char;
        tb2 : Table[0..127] of word;
```

Group: Reserved Words

4.116 task

task

Task Deklaration.

```
Task Name (MemoryArea[; Priority, RunMode]);
```

```
Task Task1 (iData);                 // default prio=5, autostart
Task Task1 (iData, 8);              // priority 8
Task Task1 (iData, resumed);        // default prio=5, automatic start
Task Task1 (iData, 8, suspended);   // prio= 8, no automatic start
```

Group: Reserved Words

4.117 then

then

Schlüsselwort für den Anweisungsteil in if Anweisung. Siehe if

Group: Reserved Words

4.118 to

to

1. bestimmt Increment der Laufvariable in for-Schleifen. Siehe for

2. Typisierung von Pointern. Siehe pointer

Group: Reserved Words

4.119 true

true

Wert einer bool'schen Konstante. Vordefiniert = $FF.

Group: Reserved Words

4.120 try

try

Beginn Implementation Teil von Exceptions.

```
Try
  StatementE..
  StatementE..
  StatementE..
except
  StatementN..
  StatementN..
EndTry;
```

Siehe auch RaiseException GetExceptResult

Group: Reserved Words

4.121 type

type

Beginn einer Typen Deklaration.

```
type    tpb = pointer to byte;
        tarr = array[2..7] of byte;
```

Group: Reserved Words

4.122 unit

unit

Definition einer Unit.

unit *Unit1;*

interface

uses *{ Liste der verwendeten Units }*

{ interface-Abschnitt }

implementation

{ implementation-Abschnitte }

initialization

{ initialization-Abschnitt }

finalization

{ finalization-Abschnitt }

end*.*

Group: Reserved Words 6

4.123 until

until

Definition der Ende Bedingung in repeat Anweisungen. Siehe repeat

Group: Reserved Words 6

4.124 userdevice

userdevice

reserviert für geplante Erweiterungen.

Group: Reserved Words 6

4.125 uses

uses

Einbindung von Units.
uses *Hello, MyMath* ***in*** *'C:\MyProg\MyMath', InitUnit;*

Group: Reserved Words

4.126 using

using

reserviert für geplante Erweiterungen.

Group: Reserved Words

4.127 val

val

reserviert für geplante Erweiterungen.

Group: Reserved Words

4.128 var

var

Beginn der Variablen Deklaration.

```
{$DATA}
Var     TrisA[$85]      : byte;         //var auf feste Adr
        Count           : byte;         //normale Deklaration
```

Group: Reserved Words

4.129 variant

variant

reserviert für geplante Erweiterungen.

Group: Reserved Words

4.130 while

while

Einleitende Statement für eine Programm-Schleife.

```
while x < 100 do
  inc (x);
endwhile;
```

Group: Reserved Words

4.131 with

with

Verkürzter Zugriff auf Records
```
with Salary do
  Individual := NewEmployee;    //Salary.Individual
  Cost := StandardRates;        //Salary.Cost
endwith;
```

Group: Reserved Words 6

4.132 word

word

1. Vordefinierter Type
16bit, 2 Bytes, 0..65535
```
var      w :       word;
 w1[24] : word;          //word an der Adresse 24…
```

2. Typecast
Wandelt das Argument in ein Word.
```
Function Word (a : type) : word;
 w:= word (x);
```

Group: Reserved Words 6

4.133 word64

word64

Vordefinierter Type. 64bit, 8 Bytes, 0..18446744073709551615
```
from System Import Word64;
...
var w64 : word64;
```

Group: Reserved Words 6

4.134 xor

xor

Operator. XOR Maske
```
a:= a xor 1;
```

Group: Reserved Words 6

5 Notizen

Index

- $ -

- 0 -

- 1 -

- 4 -

- 6 -

- 7 -

- 8 -

- A -

- B -

- C -

- D -

- E -

- F -

- G -

- H -

- I -

- K -

- L -

- M -

- N -

- O -

- P -

- R -

- S -

- T -

- U -

- V -

- W -

- X -

Sie wollen in kyrillischer Schrift, alt-griechisch oder in Hyroglyphen programmieren?
Dann benutzen Sie C.
Wenn Sie aber lesbare Programme brauchen, dann benutzen Sie AVRco und Delphi

Grombacherstr. 27
D74906 Bad Rappenau
Germany

Tel. 07268/9124-0
Fax. 07268/9124-24

Internet: www.e-lab.de
e-mail: info@e-lab.de