

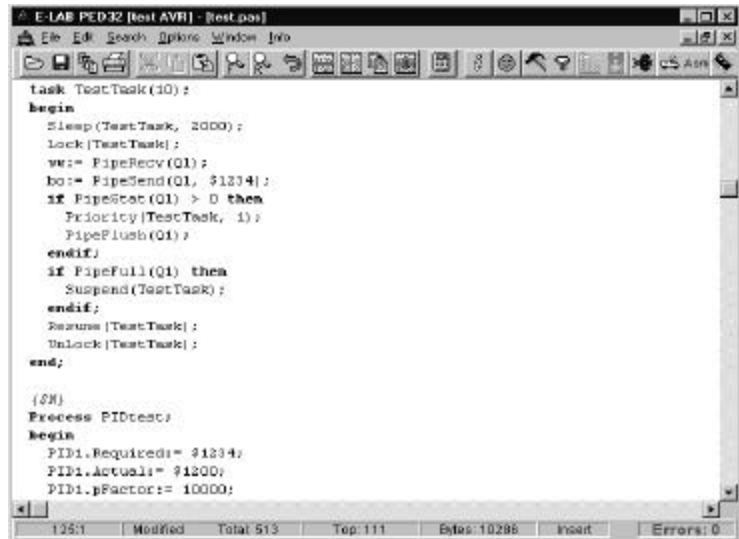
E-LAB Computers AVRco32

Pascal-scm

Features

- Pascal Compiler for Atmel's AVR 8Bit RISC cpu's, 90S2313, 90S4414 and 90S8515 etc
- Complete implementation of the Pascal standard
- Floating point: SIN, COS, PWR, PWR10, SQRT, LOG10, LOG2
- Additional strong support of bit manipulation
- Many features especially for embedded control applications
- Most of the On-Chip peripherals are implemented by comfortable library routines:
SCI, ADC, PWM, Timer
- Additional drivers implemented by software: LED7Seg Display, LCD-display, I2C-BUS, Stepper, SwitchPorts, Triggered ports, LON
- High level support of internal EEprom
- Complete **PID**-controllers impl.
- **Multiprocessing** Kernel
- Complete support of the AVR-Studio debugger on high level language. Single step on Pascal statements
- Assembler statements can be included in Pascal source
- Multi-window editor with comfortable project management
- Syntax and error highlighting
- Completely configurable environment
- Simulator included
- Application Wizzard
- Low priced
- 6 months free update
- free demo versions

Contact:
E-LAB Computers
Grombacherstr. 27
D74906 Bad Rappenau
Germany
Tel. (49) 7268 9124 0
Fax. (49) 7268 9124 24
email: info@e-lab.de
http://www.e-lab.de



Product Information

The AVRco32 is a **Multi-Task** Pascal development system for the AVR family of microcontrollers. All devices excl. 90S1200 are supported. The system was specially designed for the needs of the designer of embedded systems. Therefore many extensions for single chips are included.

AVRco32 consists of a mighty multi window editor, an Application Wizzard, the Pascal compiler, the assembler and an Incircuit-programmer.

The editor resp. IDE is completely configurable, has configurable syntax and error highlighting, unlimited filesize, multiwindow. It's heavily project oriented. Automatic reload of faulty source files. Cursor will be positioned to the incorrect syntax position. Online help of the editor functions. Context sensitive online help for the Pascal syntax. Nearly unlimited undo/redo.

The Pascal compiler supports all standard definitions and many more.
Types: Bit, Boolean, Byte, Char, Enum, Word, Integer, Pointer, String, Array, Record, Float, SysTimer, Pipe, Semaphore.

Operators: not, div, mod, and, or, xor, shl, shr, rol, ror, in

System functions: Lo, HI, Abs, Bit, Incl, Excl, Toggle, SetBit, Inc, Dec, Swap, Odd, Length, SizeOf, Delay, FillBlock, CopyBlock, WatchDog, Sleep, CpuSleep, EnableInts, DisableInts, IntToStr, ByteToStr, IntToHex, ByteToHex, FloatToStr, StrToFloat, StrToInt etc.

Statements: Begin, Return, End, If, Then, Else, Elif, Endif, Label, Goto, Case, For, While, Repeat, Loop, Type, Const, Var, Procedure, Function, With, True, False, Nil, Import.

Support of onchip peripherals: ADC, SCI, PWM, TIMER.

Software implementation of LCD-Display, triggered/debounced Ports, I2C-BUS, 7seg-Display, LAN and Stepper-Controller.

Multiprocessing-Kernel: PROCESS, TASK, SCHEDULER, PRIORITY etc.

Complete interrupt support and handling. Optional runtime errorhandling of software stack and string/array checks. Internal EEprom can be accessed as normal defined vars or as an array of bytes.

The compiler supports structured constants, forward declaration, conditional compile and also assembler statements.

The **AVR-Studio** simulator/debugger/emulator is completely integrated. Single steps and breakpoints on Pascal statements. Program variables can be examined with their Pascal names in the watch window.

Pascal-scm AVR Multitask Development

Types Boolean, Byte, Char, Bit, String, Array, Word, Integer, Enum, Procedure, Pointer, LongInt, LongWord, Float, Record, Semaphore, Pipe, SysTimer, PIDcontrol

Operants not, div, mod, and, or, xor, shl, shla, shr, shra, rol, ror, in, •, +, -

Keywords Program, From, Import, Device, Define, Const, StructConst, Var, Nil, Implementation, Procedure, Function, Process, Task, Interrupt, Trap, true, false, Begin, Return, Exit, End, ASM, EndASM, if, then, else, elsif, endif, while, repeat, break, until, Loop, ExitLoop, EndLoop, for, to, downto, by, case, exit, Label, Goto

System Library Lo, Hi, Abs, Odd, Swap, UpCase, Val, Ord, Min, Max, Random, Length, SizeOf, Incl, Excl, SetBit, Bit, Toggle, Inc, Dec, Lower, Higher, Within, Trunc, Round, Frac, Sqr, Sqrt, Pow, Pow10, Exp, Deg2Rad, Rad2Deg, ArcTan, Tangens, Sin, Cos, Log2, Log10, FillBlock, CopyBlock, StrToInt, StrToFloat, FloatToStr, IntToStr, ByteToStr, LongToStr, ByteToHex, IntToHex, LongToHex, PipeSend, PipeRecv, PipeStat, PipeFull, WaitPipe, PipeFlush, EnableInts, DisableInts, CPUSleep, Sleep, WatchDogInit, WatchDogTrig, Suspend, Resume, Lock, UnLock, Priority, Main_Priority, Schedule, WaitSema, SendSema, SemaStat, Addr, FlushBuffer, PID.Required, PID.Actual, PID.pFactor, PID.iFactor, PID.dFactor, PID.sFactor, RunTimeErr, SwitchP1, SwitchP2, Port_Stable1, Port_Stable2, Inp_Stable1, Inp_Stable2, Inp_Raise1, Inp_Raise2, mDelay, uDelay, Write, Read, LCD, LCDout, LCDinp, LCDctrl, LCDstat, RX_Buff, TX_Buff, SERinp, SERout, SERstat, ADCport, GetADC, PWMPort1, PWMPort2, PWMout1, PWMout2, I2CPort, I2Cinp, I2Cout, I2Cstat, Disp7Seg, Eeprom, Stepper.

Processes and Tasks

Pascal-scm includes a multitasking system, which is supported by an amount of functions and procedures. Up to 15 processes and tasks can be defined, which are periodically invoked by the scheduler, dependent on their priority and status.

Jobs can be done in background without intervention of the main program. The processes can communicate via pipes and semaphores. Tasks are specialized processes, which are cyclically invoked with a fixed time delay, so they can do such periodic jobs like PIDcontrols.

Multitasking is an extraordinary way to solve many problems appearing in embedded applications. The most development systems don't support it or the multitasking is sold with extra cost. Pascal-scm **includes Multi Tasking** in a streamlined way.

Free Demo Version

Prices Compiler, IDE, Assembler, 2 Manuals, Incircuit-programmer DM 940.- +MwSt \$ 590.- +ship
An additional version with linker and modules is available in 1999

6 months free updates. If expired then 30% of the actual price.

Hardware support

The most hardware functions, like SCI, ADC, PWM etc. are completely supported with functions and procedures by the system library. The internal Eeprom, if present, is treated like a normal variable, but the special access-modes of the CPU are used. So the user must not take care about it. Initialization and dealing with such CPU-parts in most cases is not an easy thing and doing this in HLL is slow and very ROM consumptive.

Several additional hardware functions, which are not supported by the CPU, like I2C, LCD, 7seg-Display, STEPPER-motor etc are also supported by the library. The PID-controller for temperature, speed or position control, which is very hated by the programmers, is also implemented. Each of this function is completely written in assembler and therefore very compact and fast. They also make the source code, because of simple function calls, shorter and readable.

Normally, these system functions are as double so fast as HLL handmade and take the half of the ROM space as their counterparts. So these features of the compiler are a big advantage against „pure“ compilers.

Compiler Switches

{\$DEFINE name}, {\$UNDEF name}, {\$IFDEF}, {\$IFNDEF}, {\$ELSE}, {\$ENDIF}

Conditional Compile switches for selecting or deselecting parts of source code.

{\$I Filename.ext}

Includes an Include-File, where "Filename" can include a file-path.

{\$J Filename.ext}

Includes an Include-File, where "Filename" shouldn't include a filepath. The "Home-Directory" of the Compiler is always used as the source filepath.

{\$DATA}, {\$IDATA}, {\$PDATA}, {\$XDATA}, {\$EEPROM}
RAM resp. variables areas. The succeeding var declarations will be placed into this area. Rem: each var can also be assigned to a physical address.

(\$R+/-) Range Check for Arrays and Strings

(\$S+/-) Stack Check

(\$NOSAVE)

Only for Interrupt procedures. No registers or data will be saved. User must save and restore them by himself.

(\$W+/-) Compiler generates warnings.