

AVR Development Tools

This section describes some of the development tools that are available for the 8-bit AVR family.

- ATMEL AVR Assembler
- ATMEL AVR Simulator
- IAR ANSI C-Compiler, Assembler, Linker, Librarian & Debugger
- ATMEL In-Circuit Emulator (ICE)
- EQUINOX Micro-Pro AVR Device Programmer

There are a lot of development tools under development, please contact ATMEL for more details.

8-Bit AVR

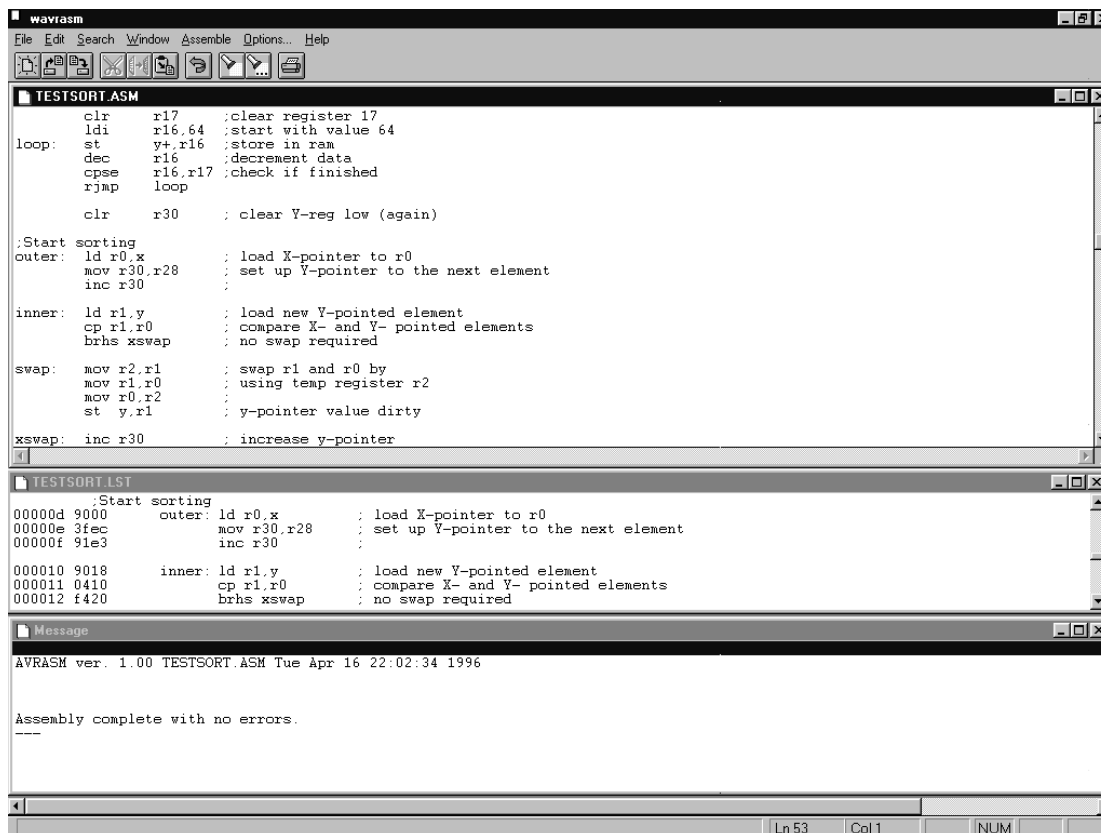
Development

Tools

ATMEL AVR Assembler

Features:

- Translates Assembler source programs into object code
- Extremely fast assembling
- Supports all the microcontrollers in the AT90S family
- Powerful Macro capabilities
- Supports all standard output formats
- Easy to use MS-Windows interface
- Editor included in MS-Windows version
- Jump to next/previous error
- Also available in MS-DOS command line version



The screenshot shows the wavrasn AVR assembler interface. It consists of three main windows:

- TESTSORT.ASM:** Contains the assembly source code for a sorting routine. Comments describe the operations, such as clearing registers, loading pointers, and performing comparisons and swaps.
- TESTSORT.LST:** Shows the listing of the assembled code, including addresses, instructions, and comments.
- Message:** Displays the output message: "AVRASM ver. 1.00 TESTSORT.ASM Tue Apr 16 22:02:34 1996" and "Assembly complete with no errors."

Powerful Macro Capabilities

The Assembler contains powerful macro capabilities, enabling the user to build a virtual instruction set which is structures of ordinary AVR instructions. For example, this macro does a 16 bit subtraction:

```

;
; SUB16 macro definition
; The macro subtracts a 16 bit constant from a register

```

```
; pair. A call to the macro is done by
; SUB16 Regh,Regl,Const
;
.MACRO SUB16                ; Macro name
    subi  $1,low($2)        ; subtract low byte
    sbci  $0,high($2)       ; subtract high byte
.ENDMACRO

; ...

; Call the macro
    ldi   r16,low(0x3400)    ; set values in registers
    ldi   r17,low(0x3400)    ;
    SUB16 r17,r16,0x23a0     ; compute 0x3400-0x23a0
```

Assembly Directives

The assembler supports a number of directives making the application development easier. In addition to the directives for macro generation and control, the assembler contains directives for:

- Including files. Included files can be nested.
- Set program origin.
- Symbol usage. The user can define symbols and labels and refer to these throughout the assembly program.
- Constant data initialization. The user can do constant initialization. Constants will be placed in the Flash program memory.
- List file control.
- Support of expressions in a C-like syntax.

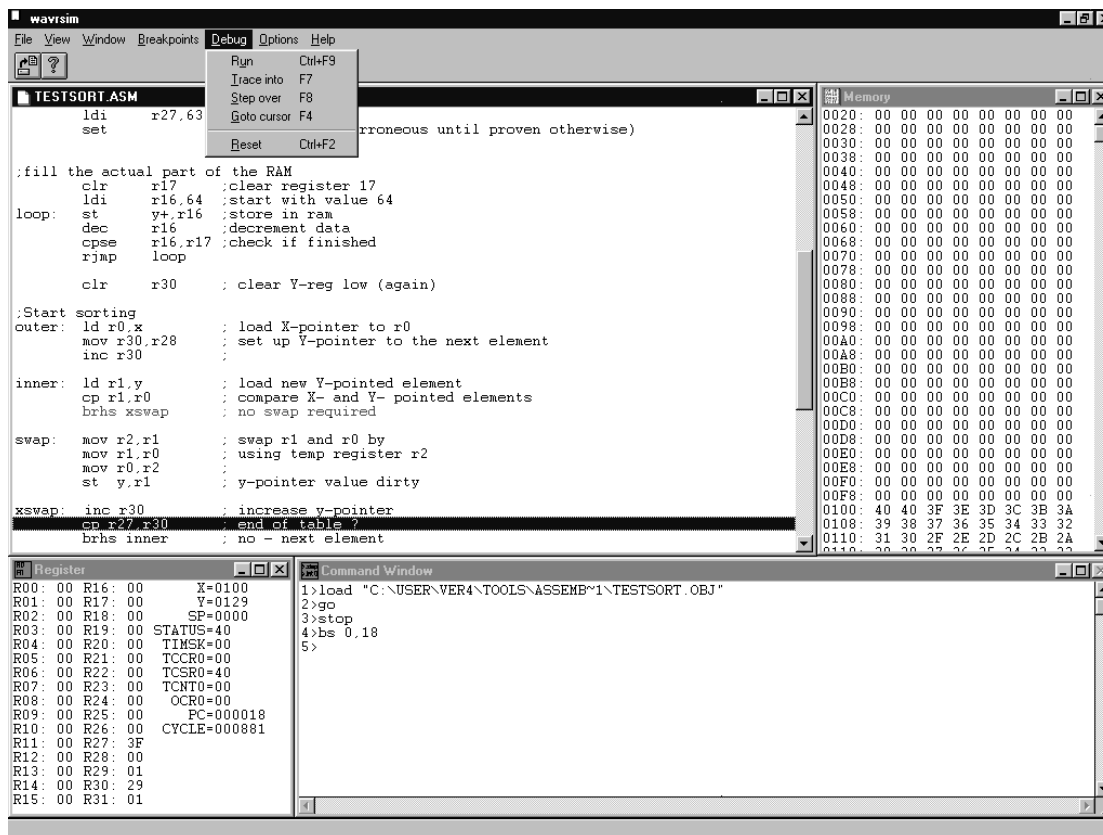
MS Windows Application

The assembler executes under the Windows environment. The Windows version can be executed under Windows 3.11, Windows 95 and Windows NT. The Windows version includes a full editor for writing assembly programs. An MS-DOS command line version is also available.

ATMEL AVR Simulator

Features:

- Supports the whole range of AT90S microcontrollers
- Assembly source level simulation
- Powerful debugging facilities
- Full support of AVR peripheral devices
- Easy to use MS-Windows interface



Debugging Facilities

The simulator has a number of functions to help the programmer to debug programs including:

- Breakpoints: Set up to 256 breakpoints in the source window, and program execution will halt upon reaching one of the breakpoints.
- Single stepping: Step through the code instruction by instruction and watch the execution.
- Step into/Trace over: Select whether calls should be traced, or if these simulation details should be omitted.
- Goto cursor: Place the cursor on an instruction, and the simulator will execute until the marked instruction is reached.
- Run from file. The user can write scripts consisting of simulator commands.
- Display of registers and memory. The user can view all memory spaces, all general purpose working registers and the registers in the I/O map. The user also has the ability to write values in these memories and registers.

- Logging. All information, including register contents, memory contents and I/O accesses can be logged for each instruction.
- The simulator holds control on the number of clock cycles elapsed.

All commands are available through a command window and through menus.

Simulation of Peripherals

The simulator supports the peripheral devices of the *AVR* microcontrollers, including:

- Interrupts. Each interrupt can be set in each cycle enabling complex combinations including nested interrupts.
- Timer/Counters. The timer/counters can also be simulated. This of course includes generating interrupts on overflows and compare matches. Free running mode is also supported.
- I/O ports. The I/O ports are implemented, giving the user the ability to communicate with the simulated programs through the ports.

Together with the powerful control mechanisms present, this makes the simulator a complete debugging tool for the *AVR* family of Enhanced RISC Microcontrollers.

MS Windows Application

The simulator is developed for execution under the Windows environment. The Windows version can be executed under Windows 3.11, Windows 95 and Windows NT.

ATMEL AVR In-Circuit Emulator

Features:

- Supports the whole range of AT90S microcontrollers
- Full visibility of all MCU resources
- Powerful breakpoint facilities
- Extensive execution control
- 32K x 96 bit wide Trace Buffer for real-time data collection
- 32-bit Time Stamp generator
- 8-Bit Event memory for event generation
- Supports 3 download modes
- Real time emulation
- Software adjustable clock speed
- Supports assembler and C source level debugging
- Supports all on-chip peripherals
- Serial- and parallel port interfacing
- Includes simple programmer
- Integrated with other AVR development tools

Full visibility

Using the emulator, the status of all resources can be monitored, and most of them can also be modified:

- The register file (R/W)
- SRAM (R/W)
- Program memory (R/W)
- EEPROM (R/W)
- Program Counter (R/W)
- I/O locations (R/W)

Powerful breakpoint facilities

The emulator incorporates powerful breakpoint facilities including:

- SRAM address breakpoint: Break when a specified address in the SRAM is read or written.
- SRAM data breakpoint: Break when a specified value is written to or read from SRAM.
- SRAM address and data breakpoint: An SRAM address breakpoint can be combined with an SRAM data breakpoint.
- Program memory address breakpoint: Break when a specified program memory address is accessed.
- Program memory data breakpoint: Break when a specified value is read from the program memory.
- Register match breakpoint: Break when a specified value is read/written from/to one of the 32 registers.
- External trigger breakpoint: Break when an external signal is rising or falling.

Extensive execution control

The emulator features extensive execution control:

- Single step execution: The emulator executes one instruction and then stops.
- Multiple step execution: The emulator executes a specified number of instructions and then stops.
- Software controlled Trace into/Step over.
- Start/Resume/Stop execution.
- Reset emulator.

Miscellaneous

- 5 trigger outputs are provided for connection to a DSO or a Logic Analyzer.
- Serial- and parallel port interface. The emulator can be connected to the PC through a standard serial- or parallel port.
- Simple programmer. A device present in the socket can be programmed from the PC or from the emulators program memory.
- In-circuit programming capabilities.
- Supports a wide range of download file formats like Intel Hex and Motorola S-Records.
- Fast download time.

EWA 90 & ICCA 90

AVR AT90S DEVELOPMENT TOOLS

EMBEDDED WORKBENCH

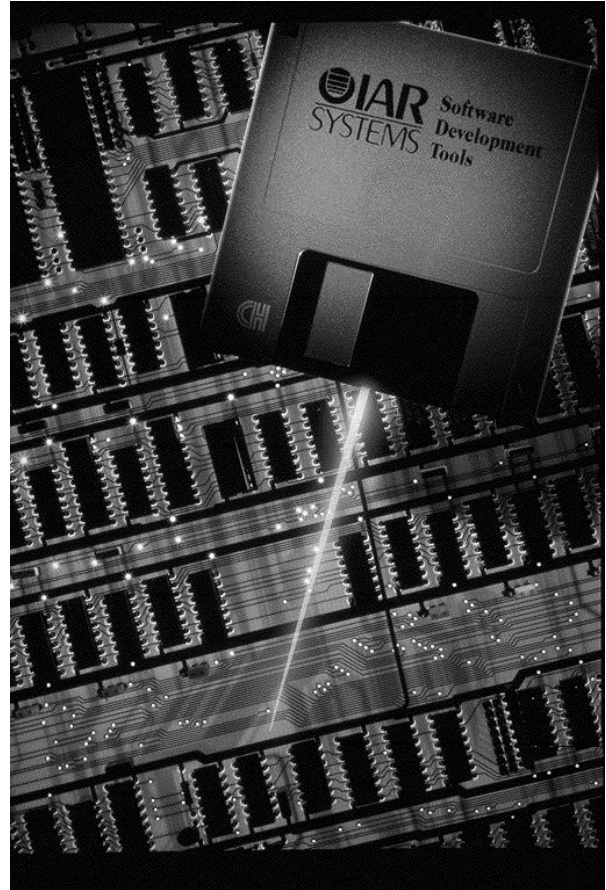
- Runs under Windows 95, NT and 3.11
- Total integration of compiler, assembler, linker and debugger
- Plug-in architecture for several IAR toolsets
- Hierarchical project presentation
- Tool options configurable on build target, group of files or on file level

C COMPILER

- Fully ANSI C compatible
- Chip specific extensions to suit development for embedded applications
- Built-in AT90S specific optimizer
- Reentrant
- Supports AT90S2312 and AT90S8414

C-SPY

- C source and Assembler level language debugger
- Powerful handling of complex breakpoints
- C like macro language
- I/O simulation
- Interrupt simulation



The IAR Embedded Workbench is a highly evolved development tool for programming embedded applications. The tool offers the choice of C to all AT90S applications, from single-chip to banked design. With its built-in chip-specific optimizer, the compiler generates very efficient, fast and reliable PROMable code for the AT90S derivatives. In addition to this solid technology, IAR also provides professional technical support. Use the IAR Embedded Workbench - and get to market faster.

AN INTEGRATED ENVIRONMENT FOR EMBEDDED PROJECT DEVELOPMENT

The Embedded Workbench, EWA90, takes full advantage of the 32 bit Windows 95 and NT environment. The toolset can also run under Windows 3.11 with the Win32s subsystem (included in the package). The EWA90 implements the intuitive Windows 95 interface with all its features.

Everything you need for programming embedded applications

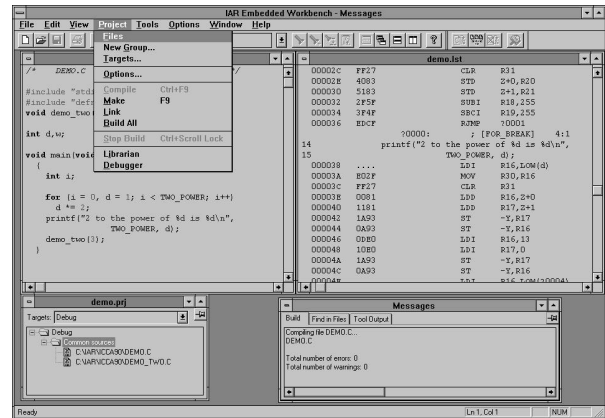
The EWA90 integrates IAR C compiler, linker, librarian, and assembler in a seamless environment with an easy-to-use project feature and option handling. Each new IAR Systems toolset for other chip families is easily integrated into the Embedded Workbench, reducing the start-up time for new targets.

Hierarchical project presentation

The project maintenance feature makes it possible to have several targets, such as a release target and a debug target, with different settings of options. Each target is built up of one or several groups which in turn are built up of one or several files. Options for the compiler and the assembler can be set on target levels, group levels or file levels, whichever is appropriate.

The Make System

The Make system automatically generates a dependency list of output files, source files and even include files. This allows the Make system to only recompile or reassemble the updated parts of the source code, which speeds up the building process.



The editor

The EWA90 offers flexibility in terms of customizable toolbar and user-defined shortcut keys. The Editor implements the basic Windows editing commands as well as extensions for C programming, such as C syntax coloring, and direct jump to context from error listing.

Extensive on-line help

The on-line help function makes it easy to quickly find specific help about the tool without leaving the Embedded Workbench, which reduces your learning time and thus time to market.

DOS user interface

The IAR C compiler is also available under DOS and is then called ICCA90. The DOS version comes with a mouse-controlled, menu driven User Interface allowing all development steps to be performed in an integrated DOS environment.

The C Compiler

A complete package including assembler, linker, librarian and run-time libraries

The IAR C compiler, the core product in the EWA90 and the ICCA90, is fully compatible with the ANSI C standard. All data types required by ANSI are supported without exception (see figure 1). Full ANSI C compatibility also means that the compiler conforms to all requirements placed by ANSI on run-time behavior, even those less known yet important requirements such as integral promotion and precision in calculating floating point.

DATA TYPE	SIZE (bytes)	VALUE RANGE
<i>sfrb</i>	1	0 to 255
<i>sfrw</i>	2	0 to 65535
<i>signed char</i>	1	-128 to +127
<i>unsigned char</i>	1	0 to 255
<i>signed short</i>	2	-32768 to +32767
<i>unsigned short</i>	2	0 to 65535
<i>signed int</i>	2	-32768 to +32767
<i>unsigned int</i>	2	0 to 65535
<i>signed long</i>	4	-2^{31} to $2^{31}-1$
<i>unsigned long</i>	4	0 to $2^{32}-1$
<i>float</i> <i>IEEE 32 bit</i>	4	$\pm 1.18E-38$ to $\pm 3.39E+38$, 7 digits
<i>pointer</i>	1-2	object address

Figure 1. Data representation supported by the IAR C compiler.

Float and *Double* are represented in the IEEE 32 precision. *Struct*, *array*, *union*, *enum*, and *bitfield* are also supported.

Reentrancy

The compiler generates fully reentrant code. Any function can be interrupted and called from the interrupting routine without the risk of corrupting the local environment of the function. This feature makes the IAR C compiler ideal to use with real time operation systems. Recursion is also supported.

Absolute read/write at C level

It is possible to access specific memory locations directly from C. The following example shows how location 10H and 11H are accessed:

```
sfrb P0IN=0x10;
sfrb P0OUT=0x11;
void read_write(char c)
{
  POUT=c;
  /*writes c to location 11H*/
  c=P0IN;
  /*reads location 10H into c*/
}
```

Built-in AT90S specific optimizer

The outstandingly and powerful optimization technique, together with a reliable and high quality program, makes the IAR C compiler unique. The AT90S architecture specific optimizer will automatically minimize the code (see figure 2).

Examples of some optimization techniques
Strength reduction. Constant Folding Short/long jump optimization. Case/Switch optimization. Register history. Peep-Hole. Additional peephole optimization. Reversing branch conditions. Removing unreachable code. Reusing duplicate code. Eliminating superfluous branches. Optimizing indeterminate storage operations.

Figure 2. The different optimization techniques that can be used by the built-in optimizer.

Depending on the application, execution speed may be more critical than smaller code. To meet this need, the compiler has the powerful feature of allowing the user to favor speed optimization over code size.

AT90S specific extensions

To ideally suit development for embedded systems, standard C needs additional functionality. IAR Systems has defined a set

of extensions to ANSI C, specific to the AT90S architecture (see figure 3). All of these extended keywords can be invoked by using the *#pragma* directive, which maintains compatibility with ANSI and code portability.

TYPE	KEYWORD	DESCRIPTION
Function	interrupt	Creates an interrupt function that is called through an interrupt vector. The function preserves the register contents and the processor status.
	monitor	Turns off the interrupts while executing a monitor function.
	C-task	Declares a function as not callable (e.g. main) to save stack.
Variable	no_init	Puts a variable in the no_init segment (battery backed RAM)
	sfrb	Maps a byte value to an absolute address
	sfrw	Maps a word to an absolute address
	tiny	Access using 8-bit address
	near flash	Access using 16-bit address Access in the program address space
Segment	codeseq	Renames the CODE segment
	constseg	Creates a new segment for constant data
	dataseg	Creates a new DATA segment (These are mostly used to place code and data sections in non consecutive address ranges)
Intrinsic	_SEI	Enable interrupt
	_CLI	Disable interrupt
	_NOP	NOP instruction
	_OPC	Inserts the opcode of an instruction into the object code
	_LPM	Returns one byte from the program address space
	_SLEEP	Enter sleep mode
	_WDR	Watch dog reset

Figure 3. IAR Systems embedded C extensions.

All these extensions, coupled with absolute read/write access, minimize the need for assembly language routines.

Register and stack usage

The compiler offers a pre-planned dynamic register allocation. This makes it possible to use all 32 general purpose registers, except Y, for expression evaluation and often used variables.

Efficient floating point

The compiler comes with full floating point support. It follows the IEEE 32-bit representation using an IAR Systems proprietary register based algorithm, which

makes floating point manipulation extremely fast.

Macro-Assembler for time-critical routines

The IAR C compiler kit comes with a new relocatable macro assembler. This provides the option of coding time-critical sections of the application in assembly without losing the advantages of the C language. The preprocessor of the C compiler is incorporated in the assembler, thus allowing header files to be shared. C include files can also be used in an assembly program. All modules written in assembly can easily be accessed from C and vice-versa, making the interface between C and assembly a straightforward process.

Powerful set of assembler directives

The assembler provides an extensive set of directives to allow total control on code and data segmentation. Directives also allow creation of multiple modules within a file, macro definitions and variable declarations.

Linker

The IAR XLINK linker supports complete linking, relocation and format generation to produce AT90S PROMable code (see figure 4). XLINK generates over 30 different formats and is compatible with most popular emulators and EPROM burners. The XLINK is extremely versatile in allocating any code or data to a start address, and checking for overflow. Detailed cross reference and map listing with segments, symbol information, variable locations, and function addresses are easily generated.

Examples of linker commands	Description
<code>-Z <i>seg_def</i></code>	Allocates a list of segments at a specific address
<code>-F <i>format_name</i></code>	Selects one of more than 30 different absolute output formats
<code>-x -l<i>file_name</i></code>	Generate a map file containing the absolute addresses of modules, segments, entry points, global/static variable, and functions
<code>-D <i>symbol=value</i></code>	Define a global symbol and equates it to a certain value

Figure 4. Example of different linker commands.

Librarian

The librarian XLIB creates and maintains libraries and library modules. Listings of modules, entry points, and symbolic information contained in every library are easily generated. XLIB can also change the attributes on a module in a file or library to be either conditionally or unconditionally loaded, i.e. loaded only if referred or loaded without being referred.

ANSI C libraries

The IAR C compiler kit comes with all libraries required by ANSI. Additionally, it comes with low level routines required for embedded systems development (see figure 5). These routines are provided in source code.

C LIBRARY FUNCTIONS
DIAGNOSTICS <assert.h> assert
CHARACTER HANDLING <ctype.h> isnum, isalnum, isalpha, iscntrl, isdigit, isgraph, islower, isprint, ispunct, isspace, isupper, isxdigit tolower, toupper
VARIABLE ARGUMENTS <stdarg.h> va_arg, va_end, va_start
NON LOCAL JUMPS <setjmp.h> longjmp, setjmp
INPUT/OUTPUT <stdio.h> getchar, gets, printf, putchar, puts, scanf, sscanf, sprintf
GENERAL UTILITIES <stdlib.h> abort, abs, atof, atoi, bsearch, calloc, div, exit, free, labs, ldiv, malloc, rand, realloc, srand, strtod, strtol, strtoul, qsort
STRING HANDLING <string.h> memchr, memcmp, memcpy, memmove, memset, strcat, strchr, strcmp, strcpy, strcspn, strerror, strlen, strncat, strncmp, strncpy, strpbrk, strrchr, strspn, strstr, strtok, strxfrm
MATHEMATICS <math.h> acos, asin, atan, atan2, ceil, cos, cosh, exp, exp10, fabs, floor, fmod, ldexp, log, log10, modf, pow, sin, sinh, sqrt, tan, tanh
PROGRAM ADDRESS SPACE ROUTINES <pgmspace.h> printf_P, puts_P, scanf_P, sprintf_P, sscanf_P, memory_P, strcmp_P, strcpy_P, strerror_P, strlen_P, strncmp_P, strncpy_P
LOW-LEVEL ROUTINES <iccbutl.h> _formatted_write, _formatted_read

Figure 5. Library functions. IAR C compiler comes with all libraries required by ANSI.

Malloc, realloc, free and heap

One powerful feature uniquely offered by the IAR C compiler kit is its ability to use *malloc*, *free*, and the flexibility of controlling the heap size. A file called *heap.c* is provided in source code, which allows the user to increase or decrease the heap.

Full, medium and small printf/scanf

By default, the compiler supports full ANSI *printf*, *sprintf*, *scanf*, and *sscanf* including floating point representation. The compiler also includes reduced versions of the *printf* and *scanf* formatters that support only the most commonly used % specifiers. This function reduces code size and increases speed. Small, medium and full *printf* formatters are available.

C-SPY

More than just a high level language debugger

The IAR C-SPY, CWA90, is a high level language debugger incorporating a complete C expression analyzer and full C-type knowledge. It combines the detailed control of code execution needed for embedded development debugging with the flexibility and power of the C language. CWA90 shows the calling stack as well as tracing on both statement and assembler levels. The source window can display C source code and mix it with assembler. There is also a "locals" window showing the auto variables and parameters for the current function.

User configurable register window

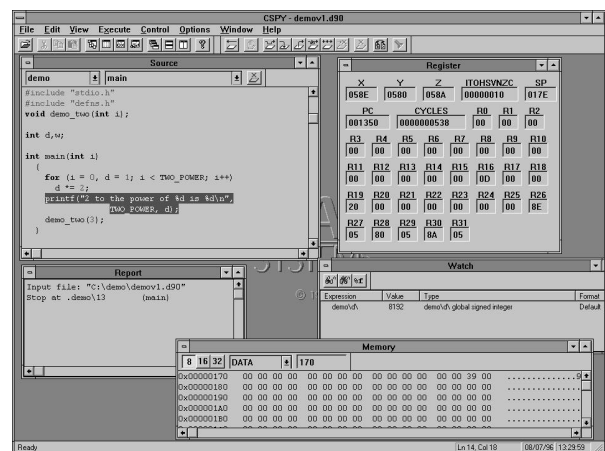
The optional CWA90 is integrated with the Embedded Workbench, implementing the intuitive Windows 95 interface with all its features. CWA90 is user-friendly with customizable toolbar, drag and drop facility and user-configurable shortcut keys. It is also configurable in the sense that the user can choose which windows to display.

Powerful breakpoint setting

CWA90 allows you to set unlimited number of breakpoints. Breakpoints can be set on C statements, assembler instructions, and on any address with an access type of *read*, *write* and *opcode fetch* or as a combination of these. The breakpoint can be extended with an optional condition. After triggering a breakpoint, any optional macro commands can be executed.

C-like macro language

A powerful, yet easy-to-use C-like macro language can tailor the environment used for debugging in the C-SPY. It includes special system macros for host file I/O simulation, reset, start up, and shut down, as well as statements such as *for* loops, *while* loops, *if* and *return*. Functions declared in the macro can contain specific C-SPY variables, both local and global.



Interrupt simulation

Interrupt simulation implements commands to launch specific interrupts at a specific cycle-count or periodically to a given cycle-interval. The interrupt simulation can also be set to generate intermittent interrupts. The simulator then uses the same algorithm as the hardware for choosing the highest priority interrupt to be executed.

I/O simulation

Breakpoint simulation and macro language allow most complex external environment to be simulated. Since I/O simulation is built up with the macro language, it is easy to customize and very easy to extend. CWA90 terminal I/O emulation offers a console window for target system I/O. This unique feature is useful for debugging embedded applications when logical flows are of interest or the target is not yet ready.

Watch points

The watch window makes it possible to watch any expression. The window itself will be updated whenever a breakpoint is triggered or a step is finished. Any variable can be modified during the execution by using specific C expressions.

Assembler low level debugger

The source window for the assembler debugger displays the assembler instructions. It has a built-in assembler and disassembler function, menu and register window, and can evaluate assembler expressions.

DOS Version

IAR C-SPY is also available in a DOS-version, the CSA90.

Summary of available AT90S Tools

- **EWA90** Windows Embedded Workbench (Including Compiler, Assembler & Linker)
- **CWA90** Debugger/Simulator for the EWA90
- **ICCA90** Integrated C Compiler for DOS (including Assembler & Linker)
- **CSA90** Debugger/Simulator for the ICCA90
- **AA90** Assembler

Support and updates

IAR Systems AT90S tool kit comes with the following benefits:

- Free telephone, fax, and email technical support.
- 90 days warranty after purchase.
- Extensive documentation including a step-by-step tutorial.

Hosts

PC: Minimum 386, with Windows95, WindowsNT 3.5 or Windows3.1x with at least 4MB RAM available for the EWA90. DOS 5.0 with at least 2MB RAM available for the ICCA90.

Contact Information:

USA

IAR Systems Inc.
One Maritime Plaza
San Francisco,
CA 94111
Tel: +1 415-765-5500
Fax: +1 415-765-5503
Email: info@iar.com

SWEDEN

IAR Systems AB
P.O. Box 23051
S-750 23 Uppsala
Tel: +46 18 16 78 00
Fax: +46 18 16 78 38
Email: info@iar.se

GERMANY

IAR Systems GmbH
Brucknerstrasse 27
D-81677 Munich
Tel: +49 89 470 6022
Fax: +49 89 470 9565
Email: info@iar.de

UK

IAR Systems Ltd.
9 Spice Court,
Ivory Square
London SW11 3UE
Tel: +44 171 924 3334
Fax: +44 171 924 5341
Email: info@iarsys.co.uk

Home Page: <http://www.iar.se>

Copyright© 1996 IAR Systems

IAR is a registered trademark of IAR Systems. Embedded Workbench, XLINK, XLIB, and C-SPY are trademarks of IAR Systems. MS-DOS and Windows are trademarks of the Microsoft Corporation. All other products are registered trademarks or trademarks of their respective owners. Product features, availability, pricing and other terms and conditions are subject to change by IAR Systems without further notice.

EQUINOX Micro-Pro AVR Device Programmer

Features

- Programs the entire range of AVR Flash-based microcontrollers
- Field programmable hardware - allows new devices to be added via a simple software update
- Fast device programming speeds due to optimized FPGA hardware for each target device
- Fast data transfer via PC parallel port
- Comprehensive front-end programming software including buffer editor
- Accepts up to 40-pin DIL devices directly without adapter

Compatible with most PCs

- Runs on IBM XT up to Pentium PC
- Compatible with desktop and Laptop PCs
- Connects to a spare PC parallel port via cable supplied
- Compatible with both standard and enhanced parallel ports

Comprehensive menu driven software:

- DOS-based - Microsoft Windows and Windows compatible
- Buffer editor - color coded displays HEX and ASCII values
- Buffer - Blank Check / Erase / View / Fill / Edit / Search / Copy / Goto
- Device - Select / Guess / Check Signature / Orientation / Auto Program / Blank Check / Erase / Program / Read / Verify / Secure
- File - Load / Save / View - Intel HEX & Binary formats
- <Auto-Program> Hot Key - Erase, program, verify & secure with one key-press
- <Load Last> Hot Key - Automatically loads last file into buffer with one key press

Micro-Pro AVR Programming System contents:

- Micro-Pro AVR device programmer
- PC Software on 3.5" HD diskette
- Power supply
- PC parallel cable
- Atmel Microcontroller Data Book

Optional programmer accessories:

- PLCC-44 to DIL-40 Package adapter (AD-PLCC44-A)
- SOIC-20 to DIL-20 Package Adapter (AD-SOIC20-A)
- In-circuit emulator / re-programming adapter for the AVR family

