

ITEL's SIDE IDE

Easy Project Management

Intel's **S**mart **I**ntegrated **D**evelopment **E**nvironment combines all of the tools needed to create and manage an embedded project. From simple control of the various tools to multiple targets and emulator communications, SIDE can handle it all. When you select your device, the options for your derivative are set for you. Does your device have multiple data pointers? No problem. Control memory allocation, Debug/Monitor configuration, and RTOS implementation all from SIDE.

Here are some of the other convenient features inside of SIDE:

- Grep
- Selected File open
- Tools Menu
- Script manager

Integrated Editor and Debugger

Source Code Editor

The SIDE source editor provides all the features that a programmer expects. From color syntax highlighting to automatic indentation for readable code, the SIDE editor has it all. Plus, the editor is available while debugging your program. Because it is integrated into the debug phase, this phase of your program development is dramatically faster.

Breakpoints

Because the editor and the simulator are integrated, you may set breakpoints at any time. While you are writing your code, you can set a breakpoint that will be in place the next time the simulator is executed. Breakpoints can be simple execution, mode complex using conditional expressions, or access specific. Additionally, you may specify that this line or symbol be on the Trace list or define the trigger.

The editor displays a useful status column to provide you with code coverage and attribute information.

Time-Saving Debug Script Language

SIDE incorporates a C-like script language, allowing you to automate every aspect of the debug and test phase of your application. It allows complete control of the simulator, collecting and saving information during an automated execution of the application with the inputs and outputs driven from the script file. Start your application in the simulator, open log files, save and time stamp data and serial communications to log files, then repeat or shut down as you need. Scripting will save you hours of work every day that you debug code.

Variables and Memory

You can simply point to a variable and display its value. Variables are classified as either local or global.

- Local variables of the active function
- Specified variables in the watch window
- Stack utilization in the Call stack view
- Memory-specific data views with memory locations independently alterable

Common Interfaces

The user interface is common among all 3 debugging tools:

- Simulator
- Monitor
- Emulator

This makes your learning curve much shorter.

C51 Compiler

The Intel compiler conforms to the ANSI standard and provides extensions to optimize its utilization for the 8051 microcontroller. This gives you the ability to write code to a well-defined standard and have complete access to all of the 8051 resources in a simple, efficient package.

Memory and SFR Access

With RC51, you have complete access to all of the memory areas within the 8051 – from internal direct memory to external off-chip memory. The keywords **sfr** and **sbit** give you access to the Special Function Registers and the bit-addressable internal memory. Because SFRs defined in standard include files derivatives with unique SFR definitions, they can be supported without explicit ITEL support. All you need to do is write them yourself from the manufacturer's data sheets.

Variables can be located at specific addresses using the **at** keyword. You can even initialize these variables at declaration. The **at** keyword can even be used to locate functions at specific locations.

RC51 has 5 separate memory models that determine the default memory location for variables. You can, of course, override the default and locate variables in any memory area you need. We also have the **generic/nogeneric** directive that allows the user to define untyped pointers to follow the memory model for pointer definition.

Code banking is supported for up to 64 banks of external memory. This allows the user to expand beyond the normal 64KB limit of the 8051. Code-banked applications are fully supported in the simulator, debugger, and emulator. No matter how large your application becomes, we can handle it.

Interrupt Functions

Interrupts can be created within RC51 using the **interrupt** keyword. At the same time, register bank selection can be specified with the **using** keyword. RC51 efficiently enters and exits interrupts making it not only convenient, but effective as well.

Flexible Pointers

Because of the Harvard memory structure of the 8051, the use of pointers can be a little more complex than for many processors. RC51 makes pointers a simple issue. By default, untyped pointers default to a 3-byte point defining the memory area and the address to which the object points. Pointers defined in this manner can point to any physical memory area. This can be altered with the **generic/nogeneric** keyword. This allows the programmer to define untyped pointers as pointers to the memory area specified by the memory model. This allows ANSI standard code and typed pointer efficiency. Alternatively, pointers can be typed and have the memory area specified. This means that pointers can now be either 1 or 2 bytes. This saves space and time, as access through a pointer can be hard coded rather than accessed through a library function.

Single- and Double-Precision Floating-Points

Like most 8051 C compilers, RC51 supports standard 32-bit IEEE floating-point math. But where the others stop, RC51 keeps going. RC51 provides functional **float**, **double**, and **long-double** data types; no one else provides this capability. **Float** data types can be defined in 32-bit IEEE-754 or BCD format. **Long** and **long-double** variables are defined in BCD and are 48 and 56 bits, respectively. This gives the programmer mantissas of up to 48 bits in length, representing numbers up to 12 digits in length. Tremendous precision capability for your analog project!

General Code Optimizations

- Constant Folding
- Global sub-expression packing
- Local sub-expression packing
- Loop rotation
- Loop optimization
- Dead code removal
- Function parameters and local variables in registers
- Parameter passing in registers
- Sub-function register optimization

8051-Specific Optimizations

- Peephole
- Absolute Jumps optimization
- Conditional Jumps optimization

Data overlaying

Factorization of complex sequences of MOV using libraries

CodeCompressor51

CodeCompressor51 is a post-linking optimizer that reduces the executable size by 18% to 20% *on average*. By optimizing the linked code, all parts of your application get optimized, not just the code generated by the compiler. This means that the C libraries and your assembly- or third-party software are all optimized. Of course, CodeCompressor51 offers you the flexibility of adjusting the scope of the optimizations from your assembly or C source code with directives that instruct CC51 to skip specified sections of code.

CodeCompressor51, available exclusively in the Enterprise Kit, performs 3 different optimization passes during execution.

- In-lining of functions called only once
- Factorization of common blocks of code through all modules
- Global peephole. Jump strength optimization.

When CC51 is finished, your code is as small as your source code allows. Just think, you can now keep your current ROM or FLASH limit, but add 18% to 20% more functionality to your project, or specify a lower cost part to do the same job. CC51 is a tremendous advantage for the 8051 developer.

Assembler Support

Any C compiler intended for the embedded market needs strong in-line assembly support. With RC51, you get two forms. The first is the industry standard directive pair ASM and ENDASM. Used in conjunction with the SRC directive, you can insert assembly source into the C source file and then have RC51 convert the C file into an assembly source file for translation by the MA51. This is a very useful form of inline assembly, but because the assembler is used as the translator, you cannot use the C source for debugging. To resolve this, we have implemented the keyword **asm{}**. This form is used for simple in-line solutions where the hex op-codes are inserted between the braces and set in the object file as is. This allows in-line assembly to be used without losing high-level language support for the C module.

So with RC51, the choice is yours: in-line assembly comes in two forms, and you can select whichever is best for you.

RC51 gives you power and flexibility in a cost-effective package.

KR51 Real-Time Kernel

KR51 is a multitasking real-time kernel that makes implementing complex, time-critical embedded projects much easier. KR51 is delivered in both library and source and is royalty free. Because it is completely implemented into SIDE, development of kernel-based applications is simple and cost effective.

XEVA Evaluation Board

8051 derivatives supported through adapters

128KB DATA

32KB ROM

Auto Baud Selection

12 MHz, upgradeable to 16 MHz

2 96 Pin connectors

4KB evaluation ANSI C compiler

ROM Monitor

Low-cost, but powerful debugging solution

51XA

8-bit mode

14.7456 MHz

Demo I/O board

2 x 16 LCD

8 LEDs

Relay

Temperature sensor

I2C peripherals PIO, RTC

External UART

External power connector

Wire wrap boards

96 Pin connector

Either XEVA connector

Replaceable for inexpensive replacement

The Choice is Obvious!

Advantages of RC51 and SIDE

- CC51, the most effective optimization available
- Complete 8051 support for all derivatives from all vendors
- C source interrupts with fast and efficient code allowing register bank selection
- Dual DPTR options for derivatives that support them
- Register variables

- Re-entrant functions and memory models to support any application need
- Code banking up to 64 banks
- Single- and double-precision floating-point support
- Real-time kernel
- Full inline assembly support
- Script language for automated debugging and testing sessions
- Common interface for simulator, monitor/debugger, and emulator
- Performance analyzer
- Code coverage
- Graphical trace viewing

Intel has been developing and selling tools for the 8051 market since 1988. We are the only tool vendor that provides every component needed for 8051 project development. From the compiler to the emulator, we can provide it all.

Now, with CC51, you can have the most efficient program at a very reasonable value, and the security of a company with over a decade of experience providing tools for the professional user.