Ingenieurbüro für Echtzeitprogrammierung

Crest-C

ANSI-C for RTOS-UH

Why C

ANSI-C is one of the most flexible and most common programming languages at all. C does not target specific areas of application, but instead provides the programmer with all tools necessary to solve the problems at hand. With the standardization of ANSI-C, featuring strict type testing and prototyping, programming in C attains the security which is mandatory for the deployment under a multitasking realtime operating system.

CREST-C

CREST-C is especially designed to support the realtime operating system RTOS-UH. An expressed goal of this development was to make a reliable programming system available, which also provides for cross-development under numerous guest operating systems like e.g. Microsoft Windows, UNIX etc. Special attention was paid on compactness and efficiency of the generated code.

CREST-C provides a hosted implementation of the ANSI-C standard X3.159-1989 (ISO/IEC 9899:1990) and is also usable as freestanding implementation.

Versatile

CREST-C allows to port lots of already available sources to RTOS-UH and to reuse existing code also under the realtime-oriented environment of RTOS-UH. The successful ports of various free sources show the reliability of the **CREST** compiler**.**

System programming

Using C, even extremely time-critical driver programming can be done using a high-level language instead of assembler. Coding of interrupt handlers and their integration within the self-configuration of RTOS-UH are completely supported.

Tasking

As hosted implementation of the C89-standard, **CREST-C** generates code following the RTOS-UH model of a shell-module on default. These modules can be called by the command interpreter and provide for parameter transfer by the command-line. Each call generates a new instance of the module, so multiple instances can act in parallel. The multi-user model of RTOS-UH is supported.

The generation of single-task programs is possible also. Tasks and subtasks can be generated at runtime, the coding of system tasks and interrupt handlers is supported.

Realtime behavior

CREST-C was developed particularly according to the specifications of the RTOS-UH operating system and provides all realtime and multitasking possibilities of the system by the means of a runtime-library. By the excellent code quality of the **CREST-C** compiler, there is no reason to code in assembler. Nevertheless, an inline-Assembler is included.

ROM ability

Aside from loadable code, **CREST-C** can generate ROM-able code. A linker can be used to bind the objects to a given base address and generate a binary image, which is directly rommable. Depending of the target processor, even position independant code can be generated.

To conserve ROM space, frequently used functions can be combined to a shared library. The linker can be instructed to bind modules to the shared library, so the library can be used simultaneously by multiple modules.

PEARL interface

To the PEARL-programmer, **CREST-C** offers the possibility to migrate to a more flexible language concept. S-Records, generated by the UH PEARL compiler, can be linked with S-Records generated by **CREST-C**. Both sides can benefit: PEARL-programs can use proven C-software, as well as C-programmers can resort to established PEARL-librarys.

Target systems

CREST-68K supports processors of the M68K-Familie; a floating point unit, if available, is supported:

- MC68000, MC68010, MC68302...
- MC68020, MC68020/MC68881, MC68030/MC68882...
- MC68040, MC68060
- CPU32, CPU32+

CREST-PPC supports processors of the power PC family:

- MPC603, MPC604, MPC750 , ...
- MPC5xx, MPC8xx, MPC82xx. ...

Libraries

ANSI-C standard libraries for the respective processor family are in the standard scope of supply. They are delivered in different translation variants, so for each application, the user can select the appropriate library.

Cross development

CREST-C is available either as generic compiler, running under RTOS-UH, or as cross-compiler for all 32-bit Microsoft Windows operating systems since Windows 95. All tools for a complete development cycle are included: the sources can be translated to either loadable or rommable objects. Testing and debugging takes place on the RTOS-UH-target, a debugger is available separately.