Using the Make Utility

The Make Utility can be used to build an application. In this case the build rules are encoded in a make file. The syntax of the make file is similar to the syntax for the GNU make.
Depending on the syntax of this make file, the make utility can:

- Rebuild the whole application, no matter if the source file is older or newer than the target file.
- Rebuild only targets when the source file is newer than the target file.

Depending on the command line syntax, the make utility can:

- Build the specified target from the make file.
- Build the first target in the make file.

Building the whole Application

If you want to build a whole application, no matter if the target are newer or older than the source file, you can write a make file with a single target and without dependency lists.

Example:

```
makeall:
  $(COMP) $(FLAGS)  calc.c
  $(COMP) $(FLAGS)  InOut.c
  $(COMP) $(FLAGS)  Terminal.c
  $(COMP) $(FLAGS)  TermPort.c
  $(LINK)  Calc.prm
```

Building only Targets, when Source Files are newer than Target Files

If you want to build only target when the source files are newer than the target files, you should associate a dependency list with each target. This dependency list contains the list of all source files, which are used to build the target file.
For an object file, this list contains the name of the ANSI C source file and the list of all header files included there.
For an executable file, this list contains the name of all the object files linked together to build the executable file.

Example:

```
.c.o:
  $(COMP) $(FLAGS) "$*.c"

Calc.abs : Calc.prm Calc.o InOut.o Terminal.o TermPort.o Start.o ansi.lib
    $(LINK)  Calc.prm

Calc.o : Calc.c InOut.h Terminal.h
InOut.o : InOut.c hidef.h InOut.h Terminal.h
Terminal.o : Terminal.c Terminal.h TermIO.h hidef.h
```

```
TermPort.o : TermPort.c TermIO.h hidef.h
```

Note:

The compiler can generate the list of dependencies for a specific source file when the option –Lm is activated.

Building a Specific Entry in the Make File

If the name of a target is specified in the command line, the Make utility only builds the specified target.

Example:

```
.c.o:
  $(COMP) $(FLAGS) "$*.c"

Calc.abs : Calc.prm Calc.o InOut.o Terminal.o TermPort.o Start.o ansi.lib
    $(LINK)  Calc.prm

Calc.o : Calc.c InOut.h Terminal.h
InOut.o : InOut.c hidef.h InOut.h Terminal.h
Terminal.o : Terminal.c Terminal.h TermIO.h hidef.h
TermPort.o : TermPort.c TermIO.h hidef.h
```

If following command line is specified, only the target “Terminal.o” is rebuild. All other object and executable file remains unchanged:

```
in> calc.mak terminal.o
```

Note:

As the target “Terminal.o” is specified with a dependency list, a new object file only is generated in one of the file listed in the dependency list is newer than the “terminal.o” file.

Building the First Target in the Make File

If the command line does not contain any target name, the make utility rebuilds the first target encountered in the make file. If the first target in the make file builds the executable file, then the whole application is rebuild. In general we recommend to specify the executable file target as first target in the make file.

Example:

Example of make file:

```
.c.o:
  $(COMP) $(FLAGS) "$*.c"

Calc.abs : Calc.prm Calc.o InOut.o Terminal.o TermPort.o Start.o ansi.lib
    $(LINK)  Calc.prm

Calc.o : Calc.c InOut.h Terminal.h
InOut.o : InOut.c hidef.h InOut.h Terminal.h
Terminal.o : Terminal.c Terminal.h TermIO.h hidef.h
TermPort.o : TermPort.c TermIO.h hidef.h
```

If following command line is specified, the target ”calc.abs” is build again.

```
in> calc.mak
```

Note:

As the targets are all specified with a dependency list, new object files only is generated in one of the file listed in the dependency list is newer than the target object file.