


CONTROLLED DISTRIBUTION ONLY IF COLOUR STAMPED

Graphical Configuration Tools

GCT Build System

CONTROLLED DISTRIBUTION LIST

GCT Design File

APPROVAL FOR DOCUMENT REVISION	Author	Checked	ELECTRONICALLY STORED DOCUMENT	
JOB TITLE	Signature	Date	DIRECTORY PATH	
Software Design Authority			/kestrel/ac00/get/docs/getbuild/getbuild.tex	
Software Team Leader			ORIGINATING DEPT: Engineering	
			CONTROLLING DEPT: Engineering	
			CONTROL SHEET	NO. of SHEETS
				16
AUTHOR: A Oliver			DOC. TYPE: Users Manual	
CONTROLLED DISTRIBUTION COPY ONLY IF COLOUR STAMPED ON CONTROL SHEET.	DOCUMENT REVISION		Graphical Configuration Tools	
	1.0		GCT Build System	
EUROTHERM CONTROLS			DOCUMENT NUMBER	SIT.
© Copyright 1985 Eurotherm Controls Ltd			HP024674C336	1

DOCUMENT REVISION HISTORY

Doc. Revision	Date	Changes
1.0	10th February, 1995	Initial Version
1.1	18th April, 1995	Updated
1.2	20th April, 1995	Added Post Build Command Documentation

Contents

1	Scope	4
2	Command Line Options	4
3	Starting GCT Build	4
4	GCT Build Configuration File: target.odt	4
4.1	Post Build Commands	5
4.1.1	%o<OptionName>	6
4.1.2	%b<OptionName>	6
5	Changes To Default.odt	6
5.1	Restoring Old Build System	6
5.2	List Of BuildOptions	7
6	Interfacing To GCT Build	8
6.1	Windows	8
6.2	Unix	8
7	GCT Build Diagnostic Output	8
7.1	Build Log	8
7.2	Command Log (Windows Only)	9

CONTROLLED DISTRIBUTION COPY
ONLY IF COLOUR STAMPED ON
CONTROL SHEET.

DOCUMENT
REVISION
1.0

Graphical Configuration Tools

GCT Build System

EUROTHERM CONTROLS

© Copyright 1995 Eurotherm Controls Ltd



DOCUMENT NUMBER
HIP024674C336

SHT.

2

8 GCT Build Operation	9
8.1 Compiled Objects	9
8.2 Librarian	9
8.2.1 Library File	9
8.2.2 Librarian for Projects	9
8.2.3 Librarian for Libraries	9
8.3 UnBuild	9
9 GCT Build Input Commands	10
9.1 Prefix	10
9.2 File Record Format	11
9.2.1 Start or End Record	11
9.2.2 Library Record	12
9.2.3 Block Record	12
9.3 Output Records	13
9.3.1 Build	13
9.3.2 UnBuild	13
9.3.3 Dates	13
9.3.4 Error Messages	14
9.3.5 Comment Message	14
9.4 Example of Dates	15
9.5 Example Of Build	15
9.6 Example Of Unbuild	16
10 GCT Build Program Structure	16
10.1 Main	16
10.2 DTB (Do The Business)	16
10.3 DDE System (Windows Only)	16

CONTROLLED DISTRIBUTION COPY
ONLY IF COLOUR STAMPED ON
CONTROL SHEET.

DOCUMENT
REVISION
1.0

Graphical Configuration Tools

GCT Build System

EUROTHERM CONTROLS

© Copyright 1995 Eurotherm Controls Ltd



DOCUMENT NUMBER
JIP024674C336

SHT.

3

1 Scope

This document is the user's guide to GCT Build, the generic build tool for GCT. The GCT Build program replaces in full the previous three utilities BUILD, UNBUILD, and DATES. Features of the GCT Build include the ability to be configured for different compilers and targets, to use DDE (Windows only) or pipes (Unix only) to communicate with GCT to reduce build times.

2 Command Line Options

The following options are available:

1. **-d** - Debug mode
2. **-v** - Verbose mode
3. **-j** - Resident/Non-Dying mode (if not set, getbuild will be instructed to die by GCT on completion of dates/build/unbuild command).
4. **-R** - Force complete rebuild of all blocks by lying about block dates.

3 Starting GCT Build

GCT Build can be invoked from any drive or directory, as it automatically changes to the correct drive and directory as it processes each command.

4 GCT Build Configuration File: target.odt

The following options can be configured:

- **ObjectExtension** - Compiled Object extension
- **CCompiler** - Command for invoking C Compiler
- **DependencyCommand** - Dependency Command - Unix only
- **CCOptionsFileCommand** - If set, specifies the compiler options/response file invoking command, eg "-f", otherwise invokes compiler with full command-line options. For compilers requiring a space between the command and the filename, underscores are translated into space, eg "f." would invoke "-f compile.cfg"
- **CFlags** - C Compiler flags for compiling standard ST blocks.
- **NonSTBodyCFlags** - C Compiler flags for compiling blocks whose body is specified in C using source.c files and "NON_ST_BODY" command in the ST. If not specified, uses CFLAGS

CONTROLLED DISTRIBUTION COPY
ONLY IF COLOUR STAMPED ON
CONTROL SHEET.

DOCUMENT
REVISION
1.0

Graphical Configuration Tools

GCT Build System

EUROTHERM CONTROLS

© Copyright 1995 Eurotherm Controls Ltd



DOCUMENT NUMBER
H1024674C336

SITT.

4

- **IncludeSTFlags** - Includes ST compiler defines from default.odt for the C compilation
- **FilesToCompile** - Optional list of files to compile. By default will compile exec.c, template.c, instance.c and resource.c (resources only). Otherwise only compiles files explicitly listed.
- **CompileFilesSeparately** - If set to "YES", then compiles files separately, otherwise will invoke compiler with list of lists.
- **RenameObjects** - Explicitly name object when compiling (only valid if CompilerFilesSeparately is "YES") using the specified command, eg "-o". For compilers requiring a space between the command and the filename, underscores are translated into space, eg "-o_" would invoke "-o filename.ObjectExtension".
- **AddIncludeSeparator** - If set "YES", adds additional directory separator after include path
- **Librarian** - Command for invoking Librarian. If none provided, will not generate a library file from the compiled objects.
- **LibrarianOptionsCommand** - If set, specifies the librarian options/response file invoking command, eg "@", otherwise invokes linker with full command-line options. For librarians requiring a space between the command and the filename, underscores are translated into space, eg "@_" would invoke "@ libopts.cfg"
- **Linker** - Command for invoking Linker. If none provided, will not attempt to link objects
- **LinkerFlags** - Linker flags for generating resource
- **LoadTaskLinkerFlags** - Linker flags for generating resource load task
- **DLLibraryImporter** - Command for invoking library importing utility. Note that this is for Windows only, where the resource is a DLL and requires importing the symbols into a library.
- **Libraries** - List of standard libraries to link into resource.
- **Function.PostBuildCommand** - List of commands (line separated) to be executed following the build of a function.
- **Function_Block.PostBuildCommand** - List of commands (line separated) to be executed following the build of a function block.
- **Program.PostBuildCommand** - List of commands (line separated) to be executed following the build of a program.
- **Resource.PostBuildCommand** - List of commands (line separated) to be executed following the build of a resource.
- **Cleanup** - If "NO", does not delete compiler and linker options/response files or exec.c files.

4.1 Post Build Commands

Post Build Commands are executed following the successful building of the particular block. A list of commands to be executed by adding "\" separators between commands.

In addition to specifying commands, variable options depending on the build can be specified.

CONTROLLED DISTRIBUTION COPY
ONLY IF COLOUR STAMPED ON
CONTROL SHEET.

DOCUMENT
REVISION
1.0

Graphical Configuration Tools

GCT Build System

EUROTHERM CONTROLS

© Copyright 1995 Eurotherm Controls Ltd



DOCUMENT NUMBER
HP024674C336

SHT.

5

4.1.1 %o<OptionName>

The command line option "%o<OptionName>" extracts the <OptionName> option from "default.odt", replacing "%o<OptionName>" with the value of this option.

If this option is not defined in "default.odt", then the <OptionName> is used.

4.1.2 %b<OptionName>

The command line option "%b<OptionName>" determines whether the <OptionName> option was defined at the start of the current build process (see "List Of BuildOptions"). If the option was defined, then "<OptionName>" is used, otherwise "No<OptionName>".

5 Changes To Default.odt

The following changes are required in default.odt to enable GCT Build to run.

- **BuildOptions** - Remove "PlainFormat"
- **BuildOptions** - Remove "NoTargetName"
- **BuildOptions** - Remove "UseOldBuildSystem" (if present)
- **BuildOptions** - Add "IncludeLibraries" or "IgnoreLibraries" depending on whether dependencies checking on libraries is required.
- **DatesCommand** - Change to "getbuild" - include appropriate path
- **BuildCommand** - Change to "getbuild" - include appropriate path
- **UnBuildCommand** - Change to "getbuild" - include appropriate path
- **BuildLogViewer** - Add name (and path) of users favorite file editor/viewer - default is notepad for windows

5.1 Restoring Old Build System

To restore the old build system, the following changes are required in default.odt.

- **BuildOptions** - Add "PlainFormat"
- **BuildOptions** - Add "UseOldBuildSystem"
- **BuildOptions** - Remove "IncludeLibraries" and Add "IgnoreLibraries"
- **DatesCommand** - Change to "dates" - include appropriate path
- **BuildCommand** - Change to "build" - include appropriate path
- **UnBuildCommand** - Change to "unbuild" - include appropriate path

CONTROLLED DISTRIBUTION COPY
ONLY IF COLOUR STAMPED ON
CONTROL SHEET.

DOCUMENT
REVISION
1.0

Graphical Configuration Tools

GCT Build System

EUROTHERM CONTROLS

© Copyright 1995 Eurotherm Controls Ltd



DOCUMENT NUMBER
HP024674C336

SHT.

6

5.2 List Of BuildOptions

Below is a summary of BuildOptions options:

- **DatesAsNeededBuildAsNeeded** (This is the default). Starting at the bottom of the dependency hierarchy Dates is called for each block. If a child needs building then don't bother calling Dates for the parent, it obviously needs building too. Call Build for every block that needs building.
- **DatesEveryBuildOnce** Call Dates for every block. Call Build ONCE at the end (Dates is expected to build up a command file for Build to execute).
- **BuildEvery** Never call Dates, call Build for every block.
- **IncludeLibraries** (This is the default). If a block depends on blocks in other libraries then build those blocks too.
- **ErrorLibraries** If a block depends on blocks in other libraries, and those blocks require building, report an error.
- **IgnoreLibraries** If a block depends on blocks in other libraries, assume that those libraries are up to date.
- **IgnoreErrors** If Dates, or Build, gets an error don't terminate the build (as we do by default) but keep going.
- **GenerateCCode** Before Build is called for a block automatically generate the 'C' code for the block (replaces the original GenerateCCode odb parameter).
- **NoTargetName** Dates, Build and Unbuild originally were not passed the name of the target that they are building for. This option is for backwards compatibility and removes the target name from the argument line. (THE DEFAULT IS TO PROVIDE THE TARGET NAME).
- **BuildBlockDependencies** By default BuildLibrary builds all blocks with their dependencies and BuildBlock just builds the block in question. This option causes build block to build dependent blocks too.
- **PlainFormat** Provided as a backward compatibility switch. When present the Object Store will
 1. Not write any start or end records to the back end utilities
 2. Invoke the build utility once per block that needs building - rather than once for all blocks
 3. Not start any record with the Stage/Util/Unit prefix
- **UseOldBuildSystem** Provided as a backward compatibility switch. When present, the object store will interface with the dates/build/unbuild utilities using files (when run under windows) instead of DDE.

CONTROLLED DISTRIBUTION COPY
ONLY IF COLOUR STAMPED ON
CONTROL SHEET.

DOCUMENT
REVISION

1.0

Graphical Configuration Tools

GCT Build System

EUROTHERM CONTROLS

© Copyright 1992 Eurotherm Controls Ltd



DOCUMENT NUMBER

11P024674C336

SHT.

7

6 Interfacing To GCT Build

6.1 Windows

Under Windows, GCT Build is a DDE server with a service name of "GCT" and topic name of "GCTBUILDSERVER". Only one client can be supported at any one time.

The following items are supported:

- "Input" - GCT Build Standard Input
- "Output" - GCT Build Standard Output
- "Error" - GCT Build Standard Error

The following DDEML services are supported:

- XTYP_WILDCONNECT
- XTYP_CONNECT
- XTYP_CONNECT.CONFIRM
- XTYP_REQUEST - for accessing "Output" and "Error"
- XTYP_POKE - for writing "Input"
- XTYP_DISCONNECT

A GCT Build DDE client will cause GCT Build to die if an "ABORT" input command is sent, or the client disconnects from the server.

6.2 Unix


Under Unix, GCT Build receives and sends commands using pipes.

7 GCT Build Diagnostic Output

Besides communicating with GCT using DDE or Pipes, GCT Build generates a number of files.

7.1 Build Log

The output from each spawned command (compiler, librarian, link, etc) is logged to a file called "buildlog.txt" in the root directory of GCT. This enables the user to inspect the build output on completion of a build. This file is deleted each time GCT Build is started.

CONTROLLED DISTRIBUTION COPY ONLY IF COLOUR STAMPED ON CONTROL SHEET.	DOCUMENT REVISION	Graphical Configuration Tools	
	1.0	GCT Build System	
EUROTHERM CONTROLS <small>© Copyright 1996 Eurotherm Controls Ltd</small>		DOCUMENT NUMBER	SHT.
		III'024674C336	8

7.2 Command Log (Windows Only)

If the Debug flag is specified on the command line, then all commands and responses are logged to a file called "gctbuild.cnd" in the root directory of GCT.

8 GCT Build Operation

8.1 Compiled Objects

Compiled objects are stored in a object directory under the project's directory as follows:

<Project Directory Path>/lib/<Target Name>/objects

The objects are name using one character to define the type ("c" for exec, "i" for instance, or "t" for template) followed by the 7 character function block directory name.

Resource function block objects remain in the function block directory itself.

8.2 Librarian

8.2.1 Library File

Library files are always stored in the following directory:

<Project/Library Directory Path>/<Target Name>

8.2.2 Librarian for Projects


When configured to create a library, the library file is only created (not updated) when a resource function block is being compiled - this assumes that all dependent function blocks are compiled before. Then, ALL objects within the target's object directory are added to the library file.

8.2.3 Librarian for Libraries

When configured to create a library, the library file is updated after each function block is compiled.

8.3 UnBuild

During an unbuild, the function block object files (exec, template, instance) are deleted from the object directory (for resource function block, from the function block directory itself) and removed from the librarian (if one exists).

CONTROLLED DISTRIBUTION COPY ONLY IF COLOUR STAMPED ON CONTROL SHEET.	DOCUMENT REVISION 1.0	Graphical Configuration Tools GCT Build System	
EUROTHERM CONTROLS © Copyright 1995 Eurotherm Controls Ltd		DOCUMENT NUMBER HIP024674C336	SHT. 9

9 GCT Build Input Commands

9.1 Prefix

Each input record is on a separate line and starts with a prefix of the form Stage/Util/Unit, where, Stage is the stage of the utility operation. It takes the following values

- S - the start of the utility. There is always one 'S' record at the start of a utility
- L - operation on an individual included library (i.e. one represented by library.st)
- B - operation on an individual block
- E - the end of the utility. There is always one 'E' record at the end of a utility

Util indicates the utility being invoked. It takes the following values:

- D - the dates utility
- B - the build utility
- U - the unbuild utility

Unit is the unit on which the user is operating. It takes the following values

- L - the user has performed an operation on a library (e.g. Build Library)
- P - the user has performed an operation on a library (e.g. Build Project)
- B - the user has performed an operation on an individual block (e.g. Build Block)

Thus the prefix SDB is the Start record of the Dates utility for a build Block operation and BUL is a Block record of the UnBuild utility for an unbuild Library operation.

CONTROLLED DISTRIBUTION COPY
ONLY IF COLOUR STAMPED ON
CONTROL SHEET.

DOCUMENT
REVISION
1.0

Graphical Configuration Tools

GCT Build System

EUROTHERM CONTROLS

© Copyright 1995 Eurotherm Controls Ltd



DOCUMENT NUMBER
IIP024674C336

SHT.

10

9.2 File Record Format

9.2.1 Start or End Record

For a project or library operation (Unit is 'L' or 'P'):-

<PREFIX> <DIR_NAME> <BLOCKS_DIR> <LIB_NAME> <TARGET_NAME> <OPTIONS>

where,

- <PREFIX> is S/E D/B/U U/L
- <DIRNAME> is the hierarchic path to the project or library directory.
- <BLOCKSDIR> is the relative directory path to the directory / file holding the block definitions (usually blocks, but library.st for an Included library)
- <LIBNAME> is the name of the project or library.
- <TARGETNAME> is the name of the current build target
- <OPTIONS> are the user specified build options (combination of odb options and those passed across the OSI to the back end operation).

For a block operation (Unit is 'B'):-

<PREFIX> <DIR NAME> <BLOCK_NAME> <BLOCK_TYPE> <TARGET_NAME> <OPTIONS>

where,

- <PREFIX> is S/E D/B/U B
- <DIR_NAME> is the hierarchic path to the block directory.
- <BLOCKNAME> is the name of the block being built
- <BLOCKTYPE> is the type of block (e.g. PROGRAM, FUNCTION-BLOCK, etc).
- <TARGETNAME> is the name of the current build target
- <OPTIONS> are the user specified build options (combination of odb options and those passed across the OSI to the back end operation).

CONTROLLED DISTRIBUTION COPY
ONLY IF COLOUR STAMPED ON
CONTROL SHEET.

DOCUMENT
REVISION
1.0

Graphical Configuration Tools

GCT Build System

EUROTHERM CONTROLS

© Copyright 1995 Eurotherm Controls Ltd



DOCUMENT NUMBER
IIP024674C336

SHT.

11

9.2.2 Library Record

For the Dates utility:-

<PREFIX> <DIR_NAME> library.st LIBRARY <TARGET_NAME> <CHILD_DATES>

where,

- <PREFIX> is LD U/L
- <DIRECTORYNAME> is the hierarchic path to the library directory.
- <TARGETNAME> is the name of the current build target
- <CHILDDATES> are the dates of the library's dependent library's most recent build time, source modification and interface modification. The dates are those returned by the Dates utility - see the output record format.

For the Build and UnBuild utilities:-

<PREFIX> <DIR_NAME> library.st LIBRARY <TARGET_NAME>

where,

- <PREFIX> is L B/U U/L

9.2.3 Block Record

For the Dates utility:-

<PREFIX> <DIR_NAME> <BLOCK_NAME> <BLOCK_TYPE> <TARGET_NAME>
<CHILD_DATES>

where,

- <PREFIX> is BD U/L
- <DIRECTORYNAME> is the hierarchic path to the block directory.
- <BLOCKSNAME> is the name of the block being built
- <BLOCKTYPE> is the type of block (e.g. PROGRAM, FUNCTION BLOCK, etc).
- <TARGETNAME> is the name of the current build target

CONTROLLED DISTRIBUTION COPY
ONLY IF COLOUR STAMPED ON
CONTROL SHEET.

DOCUMENT
REVISION
1.0

Graphical Configuration Tools

GCT Build System

EUROTHERM CONTROLS

© Copyright 1995 Eurotherm Controls Ltd



DOCUMENT NUMBER
IIP024674C336

SHT.

12

- <CHILDDATES> are the dates of the block's dependent block's most recent build time, source modification and interface modification. The dates are those returned by the Dates utility - see the output record format.

For the Build and UnBuild utilities:-

<PREFIX> <DIR.NAME> <BLOCK.NAME> <BLOCK.TYPE> <TARGET.NAME>

where,

- <PREFIX> is B B/U U/L

9.3 Output Records

9.3.1 Build

All output is regarded as comments to be printed on the standard output channel (GCT build window).

9.3.2 UnBuild

All output is regarded as comments to be printed on the standard output channel (GCT build window).

9.3.3 Dates

If a build is required, the following output is produced:


B [<date1> [<date2>]]

this indicates that the block needs building. The optional dates are intended to provide further information that the object store cannot derive itself.

- <date1> is the date that the block's source(s) last changed
- <date2> is the date that the block's interface last changed.

These two dates (which default to 0xFFFFFFFF) are returned by the object store to future invocations of Dates in the input records for parent blocks.

If Don't Build is required, the following output is produced:

CONTROLLED DISTRIBUTION COPY ONLY IF COLOUR STAMPED ON CONTROL SHEET.	DOCUMENT REVISION 1.0	Graphical Configuration Tools GCT Build System	
EUROTHERM CONTROLS © Copyright 1995 Eurotherm Controls Ltd		DOCUMENT NUMBER HP024674C336	SIT. 13

D <date1> [<date2> [<date3>]]

this indicates that the block doesn't need building.

- <date1> is the last build date for this block
- <date2> is the date that the block's source(s) last changed (defaults to <date1>)
- <date3> is the date that the block's interface last changed (defaults to <date1>)

9.3.4 Error Messages

The output:

B <error message>

indicates an error. The message is written to the error window.

9.3.5 Comment Message


The output

<Comment>

indicates an informative comment. The message is written to the output window.

The presence of any error output will indicate a failure which will terminate the build process (unless the option IgnoreErrors is set).

The format of a date is unsigned numeric - its exact meaning is up to the build system. The object store only uses the numeric value to determine if a block is older (lower numeric value) or younger (higher numeric value). The child date passed into the Dates program is the date given for the child block most recently (highest value) built. This value is derived from the results of calling Dates for the child blocks. A 'B' o32put record without a date is interpreted as the maximum date (2³¹ or 0xFFFFFFFF) - i.e. in the future.

CONTROLLED DISTRIBUTION COPY ONLY IF COLOUR STAMPED ON CONTROL SHEET.	DOCUMENT REVISION 1.0	Graphical Configuration Tools	
		GCT Build System	
EUROTHERM CONTROLS © Copyright 1995 Eurotherm Controls Ltd		DOCUMENT NUMBER	SHEET
		HP024674C336	14

9.4 Example of Dates

- SDP D:/gct/exec/project/EXTRUDER blocks EXTRUDER Windows DatesAsNeededBuildAsNeeded IncludeLibraries GenerateCCode BuildBlockDependencies
- LDP D:/gct/exec/library/TASKS TASKS LIBRARY Windows 0 0 0
- D 18352 12091 -13978
- SDL D:/gct/exec/library/GENIO blocks GENIO Windows
- RDP D:/gct/exec/library/GENIO/BLOCKS/DIGITAL DigitalOut FUNCTION_BLOCK Windows 0 0 0
- D 18352 12091 -13978
- EDL D:/gct/exec/library/GENIO blocks GENIO Windows
- BDP D:/gct/exec/project/EXTRUDER/BLOCKS/HOPPERC HopperCtrl FUNCTION_BLOCK Windows 18352 12091 4294953318
- B 25974 11996
- BDP D:/gct/exec/project/EXTRUDER/BLOCKS/RACK Rack PROGRAM Windows 0 0 0
- B -25632 11993
- BDP
D:/gct/exec/project/EXTRUDER/BLOCKS/EXTRUDE Extrud RESOURCE Windows 4294941664 4294941664 4294941664
- B -13034 12080
- EDP D:/gct/exec/project/EXTRUDER blocks EXTRUDER Windows

9.5 Example Of Build

- SBP D:/gct/exec/project/EXTRUDER blocks EXTRUDER Windows DatesAsNeededBuildAsNeeded IncludeLibraries GenerateCCode BuildBlockDependencies
- BBP D:/gct/exec/project/EXTRUDER/BLOCKS/HOPPERC HopperCtrl FUNCTION_BLOCK Windows
- Built block 'HopperCtrl'
- BBP D:/gct/exec/project/EXTRUDER/BLOCKS/RACK Rack PROGRAM Windows
- Built block 'Rack'
- BBP D:/gct/exec/project/EXTRUDER/BLOCKS/EXTRUDE Extrud RESOURCE Windows
- Built resource 'Extrud'
- EBP D:/gct/exec/project/EXTRUDER blocks EXTRUDER Windows

CONTROLLED DISTRIBUTION COPY
ONLY IF COLOUR STAMPED ON
CONTROL SHEET.

DOCUMENT
REVISION
1.0

Graphical Configuration Tools

GCT Build System

EUROTHERM CONTROLS

© Copyright 1995 Eurotherm Controls Ltd



DOCUMENT NUMBER
HP024674C336

SHEET

15

9.6 Example Of Unbuild

- SUP D:/get/exec/project/EXTRUDER blocks EXTRUDER Windows DatesAsNeededBuildAsNeeded IncludeLibraries GenerateCCode BuildBlockDependencies
- BUP D:/get/exec/project/EXTRUDER/BLOCKS/HOPPERC HopperCtrl FUNCTION.BLOCK Windows
- # Unbuilt block 'HopperCtrl'
- BUP D:/get/exec/project/EXTRUDER/BLOCKS/EXTRCON ExtrControl PROGRAM Windows
- # Unbuilt block 'ExtrControl'
- BUP D:/get/exec/project/EXTRUDER/BLOCKS/EXTRUDE Extrud RESOURCE Windows
- # Unbuilt block 'Extrud'
- EUP D:/get/exec/project/EXTRUDER blocks EXTRUDER Windows

10 GCT Build Program Structure

10.1 Main

Both the Unix and Windows main program consist of a loop where execution waits for incoming messages to become available from the input pipe (Unix) or dde (Windows) before passing execution into the "dtb" function (Do The Business). If the input pipe is closed or the dde client disconnects, the loop is exited and the program terminates.

10.2 DTB (Do The Business)

This function processes the input message, determining which function is required (Dates, Build, or Unbuild), whether it is a Start (causes date structures to be initialized with the appropriate target date), Library (only processes Dates commands), Block (appropriately passes execution to do_dates, do_build, or do_unbuild) or End command (cleans structures).


The do_dates, do_build, and do_unbuild functions perform the appropriate processing before returning.

10.3 DDE System (Windows Only)

Messages to and from GCT Build are passed into FIFO buffers. These buffers are handled by the FIFOBuffer class.

The interface to the DDE windows system is handled by the ddeBuildServer class, which is derived from the ddeConnectServer class (see GCT DDE Server). This class handles reading and writing to the FIFO buffers, DDE Callbacks, DDE message loops, and Windows message loops.


→ ○ ←

CONTROLLED DISTRIBUTION COPY ONLY IF COLOUR STAMPED ON CONTROL SHEET.	DOCUMENT REVISION	Graphical Configuration Tools	
	1.0	GCT Build System	
EUROTHERM CONTROLS <small>© Copyright 1995 Eurotherm Controls Ltd</small>		DOCUMENT NUMBER	SHT.
		HP024674C336	16

CONTROLLED DISTRIBUTION ONLY IF COLOUR STAMPED

GCT

GCT Build System

APPROVAL FOR DOCUMENT REVISION	Author	Checked	ELECTRONICALLY STORED DOCUMENT	
JOB TITLE	Signature	Date	DIRECTORY PATH /osprey4/gct/doc/specs/buildsys/buildsys.tex	
			ORIGINATING DEPT: ENGINEERING	
			CONTROLLING DEPT: ENGINEERING	
			CONTROL SHEET	NO. of SHEETS 12
AUTHOR: Ian Hughes			DOC. TYPE: Software Design Specification	
CONTROLLED DISTRIBUTION COPY ONLY IF COLOUR STAMPED ON CONTROL SHEET.	DOCUMENT REVISION 4	GCT		
		GCT Build System		
EUROTHERM CONTROLS <small>© Copyright 1994 Eurotherm Controls Ltd</small>			DOCUMENT NUMBER HP024674C308	SHT. 1

2.1 Project and Library Directories

The differences between projects and libraries are relatively minor in internal terms (only a project can hold CONFIGURATIONs and RESOURCEs) so they will be considered here together.

The top level directory (project or library) acts as a place holder for individual project and library directories. Currently a project/library name is also the directory name and so is limited to 8 characters (for portability to DOS). It would be possible eventually to have a `dir.tbl.tab` type file in each of project and library to act as a directory mapping longer project/library names (say up to 20 characters) onto directory names (8 characters).

Inside a project/library directory are:-


- A mapping file (`dir.tbl.tab`) that maps CDL block (Function Block, Program, Resource, etc.) names (up to 20 characters) onto directory names (7 characters). Directory names are limited to 7 characters so that further unique names can be derived from it and still fit the DOS 8 character limit.
- A place holder directory (blocks) for individual CDL block directories.
- A directory per target type (if required by the target build system) for holding target specific build files (e.g. `lib.a`).
- An options database section file (`library.odt`) holding the current build options for the library, the library search list, etc..

Note that there is a particular form of library (an *Included Library*) that is used to store user defined types and function prototypes. In an Included Library there is a single source file (`library.st`) held in the top level directory and containing all the type definitions and prototypes.

Inside an individual CDL block directory are:-

- `source.st` - the CDL source file which holds at least the wrapper definition and the editor attribute.
- Other files which depend on the block type, the targets and the build processes used to build the block.

Figure 1 illustrates the file and directory structure.

CONTROLLED DISTRIBUTION COPY ONLY IF COLOUR STAMPED ON CONTROL SHEET.	DOCUMENT REVISION 4	GCT	
		GCT Build System	
EUROTHERM CONTROLS © Copyright 1994 Eurotherm Controls Ltd		DOCUMENT NUMBER	
		HP024674C308	SHT. 3

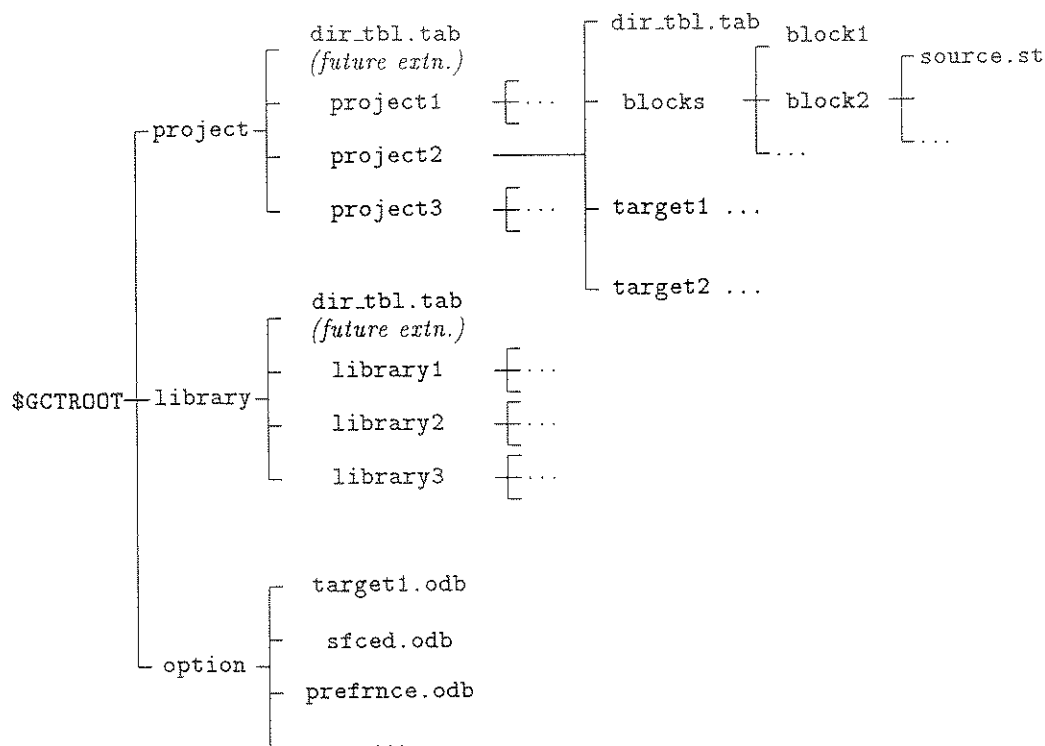


Figure 1: Directory and File Structure


3 The Build Process

The build process is handled by the object store — this means that the editors do not need to concern themselves with the mechanics of building. The *osi* (Object Store Interface) offers the following interfaces for building :-

- `osiBlock::Build` — builds an individual block.
- `osiLibrary::Build` — builds all the blocks in a project or library.
- `osiBlock::UnBuild` — unbuilds an individual block (forces the block to be rebuilt before use).
- `osiLibrary::UnBuild` — unbuilds all blocks in a project or library (forces all blocks to be rebuilt before use).
- `osiBlock::ListBlockDependencies` — lists which blocks a given block is dependent on, to as many levels as requested.

GCT supports:-

- Different target build systems. The same source block may be targetted at multiple products with very different run time architectures.
- Different source files. Some editors (E.g. a graphics editor) will create files that GCT knows nothing about.

CONTROLLED DISTRIBUTION COPY ONLY IF COLOUR STAMPED ON CONTROL SHEET.	DOCUMENT REVISION 4	GCT GCT Build System	
EUROTHERM CONTROLS © Copyright 1994 Eurotherm Controls Ltd		DOCUMENT NUMBER HP024674C308	SHT. 4

The build process involves several stages. The exact use of these stages can be fine tuned for a particular build system as described later.

1. Identify all the blocks that the 'to be built' block is dependent on (to a depth determined by the `CompilerOptions` option).
2. Invoke command files (scripts, batch files, or executables) to identify which of the above blocks require re-building. N.b. The object store cannot determine this as back-ends may create other source/output files that are unknown to the object store.
3. Invoke command files to perform the builds of the individual blocks.

The following command files are required to support the build process for any given type of block:-


1. *Dates* — to identify the latest source date and the latest build date. The output is appended to a "results" file in a platform dependent manner.
2. *Build* — to perform the necessary operations for building the block. This command file must support building the block type for all the targets supported by the type. The command file can assume that the ST has been validated.
3. *UnBuild* — remove all but source files.

This build process can be fine tuned by adjusting target specific options database parameters. The following parameters affect the build process:-

1. `<target>:<block type>.CompilerOptions` — including the `-GI` option for a block type ensures that the full block hierarchy is read into the object store when the block is opened. Otherwise only one level of dependencies are identified and the build process will not identify all dependent blocks that may require building. The penalty for setting `-GI` is the increased time taken to open the block. `<block type>` may be `RESOURCE`, `PROGRAM` or `FUNCTION BLOCK` — normal practice would be to set the `-GI` flag for `RESOURCES` only. It should be noted that the `-GI` flag is set automatically by the object store whenever a `PROGRAM` is built in a Build Library operation. (*There are inconsistencies here that require resolution — e.g. what about instance.c for a function block at the RESOURCE level?*)
2. `<target>:BuildOptions` — this option fine tunes the build process itself. It takes a value made up of a list of options:-

```
<target>:BuildOptions:= [ DatesAsNeededBuildAsNeeded |
                        DatesEveryBuildOnce |
                        BuildEvery ]
                        [ IncludeLibraries |
                        ErrorLibraries |
                        IgnoreLibraries ]
                        [ IgnoreErrors ]
                        [ GenerateCCode ]
                        [ NoTargetName ]
                        [ BuildBlockDependencies ]
                        [ PlainFormat ]
```

where,

CONTROLLED DISTRIBUTION COPY ONLY IF COLOUR STAMPED ON CONTROL SHEET.	DOCUMENT REVISION 4	GCT GCT Build System	
EUROTHERM CONTROLS <small>© Copyright 1994 Eurotherm Controls Ltd</small>		DOCUMENT NUMBER HP024674C308	SHT. 5

DatesAsNeededBuildAsNeeded (This is the default). Starting at the bottom of the dependency hierarchy *Dates* is called for each block. If a child needs building then don't bother calling *Dates* for the parent, it obviously needs building too. Call *Build* for every block that needs building.

DatesEveryBuildOnce Call *Dates* for every block. Call *Build* ONCE at the end (*Dates* is expected to build up a command file for *Build* to execute).

BuildEvery Never call *Dates*, call *Build* for every block.

IncludeLibraries (This is the default). If a block depends on blocks in other libraries then build those blocks too.

ErrorLibraries If a block depends on blocks in other libraries, and those blocks require building, report an error.

IgnoreLibraries If a block depends on blocks in other libraries, assume that those libraries are up to date.

IgnoreErrors If *Dates*, or *Build*, gets an error don't terminate the build (as we do by default) but keep going.

GenerateCCode Before *Build* is called for a block automatically generate the 'C' code for the block (replaces the original GenerateCCode odb parameter).

NoTargetName *Dates*, *Build* and *Unbuild* originally were not passed the name of the target that they are building for. This option is for backwards compatability and removes the target name from the argument line. (THE DEFAULT IS TO PROVIDE THE TARGET NAME).

BuildBlockDependencies By default BuildLibrary builds all blocks with their dependencies and Build-Block just builds the block in question. This option causes build block to build dependent blocks too (currently limited to a depth of ONE).

PlainFormat Provided as a backward compatability switch. When present the Object Store will

- Not write any start or end records to the back end utilities
- Invoke the build utility once per block that needs building - rather than once for all blocks
- Not start any record with the *Stage/Util/Unit* prefix


During the build process a CallBack will be invoked at various stages:-

1. To report the blocks that require building (as long as the **BuildEvery** option is not set).
2. Report any dependent libraries that are unbuilt (the build process can optionally be aborted at this stage). (*not yet implemented*)
3. Report progress on the building of blocks (the build process can optionally be aborted between building one block and the next) - but only if PlainFormat is set (otherwise it is the build utility's responsibility to detect the error and stop if appropriate).
4. Return any output or errors reported by a build command file (build will automatically be aborted on errors).

4 Interface To Build Back-End

The interface the the build support commands (*Dates*, *Build*, and *UnBuild*) is as follows:-

1. (Under Windows-3) Execute the build support programs using 'WinExec' - the 'program' could then be a DOS executable, a WINDOWS executable or a .BAT file (I believe)

CONTROLLED DISTRIBUTION COPY ONLY IF COLOUR STAMPED ON CONTROL SHEET.	DOCUMENT REVISION 4	GCT GCT Build System	
EUROTHERM CONTROLS © Copyright 1994 Eurotherm Controls Ltd		DOCUMENT NUMBER HP024674C308	SHT. 6

2. (Under Windows-3) Pass all input to the program *in a file* of predefined name (GCTIN) in the current working directory (the directory of the block / library being built).
3. (Under Windows-3) Expect output *in a file* in the same directory (GCTOUT), and errors in a separate file (GCTERR); with a separate file (GCTDONE) created purely to signify that the program is finished.
4. (Under Unix) the same principle is used except that the files are replaced by `stdin`, `stdout` and `stderr` pipes
5. The input to the *Dates* program will be a file containing the blocks to test (one per line). The output will be a file containing the blocks that need making (one per line).

Each input record is on a separate line and starts with a prefix of the form *Stage/Util/Unit*, where,

Stage is the stage of the utility operation. It takes the following values

- S - the start of the utility. There is always one 'S' record at the start of a utility
- L - operation on an individual included library (i.e. one represented by `library.st`)
- B - operation on an individual block
- E - the end of the utility. There is always one 'E' record at the end of a utility

Util indicates the utility being invoked. It takes the following values

- D - the dates utility
- B - the build utility
- U - the unbuild utility

Unit is the unit on which the user is operating. It takes the following values

- L - the user has performed an operation on a library (e.g. Build Library)
- P - the user has performed an operation on a library (e.g. Build Project)
- B - the user has performed an operation on an individual block (e.g. Build Block)


Thus the prefix

SDB

is the **S**tart record of the **D**ates utility for a build **B**lock operation and

BUL

is a **B**lock record of the **U**nBuild utility for an unbuild **L**ibrary operation.

CONTROLLED DISTRIBUTION COPY ONLY IF COLOUR STAMPED ON CONTROL SHEET.	DOCUMENT REVISION 4	GCT GCT Build System	
EUROTHERM CONTROLS © Copyright 1994 Eurotherm Controls Ltd			DOCUMENT NUMBER HP024674C308 SHT. 7

4.1 Input File Record Format

4.1.1 Start or End Record

For a project or library operation (Unit is 'L' or 'P'):-

<PREFIX> <DIRECTORY_NAME> <BLOCKS_DIR> <LIB_NAME> <TARGET_NAME> <OPTIONS>

where,

<PREFIX> is S|E D|B|U U|L

<DIRECTORY_NAME> is the hierarchic path to the project or library directory.

<BLOCKS_DIR> is the relative directory path to the directory / file holding the block definitions (usually blocks, but library.st for an Included library)

<LIB_NAME> is the name of the project or library.

<TARGET_NAME> is the name of the current build target

<OPTIONS> are the user specified build options (combination of odb options and those passed across the OSI to the back end operation).

For a block operation (Unit is 'B'):-

<PREFIX> <DIRECTORY_NAME> <BLOCK_NAME> <BLOCK_TYPE> <TARGET_NAME> <OPTIONS>

where,

<PREFIX> is S|E D|B|U B

<DIRECTORY_NAME> is the hierarchic path to the block directory.

<BLOCK_NAME> is the name of the block being built

<BLOCK_TYPE> is the type of block (e.g. PROGRAM, FUNCTION_BLOCK, etc).

<TARGET_NAME> is the name of the current build target

<OPTIONS> are the user specified build options (combination of odb options and those passed across the OSI to the back end operation).

CONTROLLED DISTRIBUTION COPY
ONLY IF COLOUR STAMPED ON
CONTROL SHEET.

DOCUMENT
REVISION

4

GCT

GCT Build System

EUROTHERM CONTROLS

© Copyright 1994 Eurotherm Controls Ltd



DOCUMENT NUMBER
HP024674C308

SHT.

4.1.2 Library Record

For the *Dates* utility:-

<PREFIX> <DIRECTORY_NAME> library.st LIBRARY <TARGET_NAME> <CHILD_DATES>

where,

<PREFIX> is LD U|L

<DIRECTORY_NAME> is the hierarchic path to the library directory.

<TARGET_NAME> is the name of the current build target

<CHILD_DATES> are the dates of the library's dependent library's most recent build time, source modification and interface modification. The dates are those returned by the *Dates* utility - see the output record format.

For the *Build* and *UnBuild* utilities:-

<PREFIX> <DIRECTORY_NAME> library.st LIBRARY <TARGET_NAME>

where,

<PREFIX> is L B|U U|L

4.1.3 Block Record

For the *Dates* utility:-

<PREFIX> <DIRECTORY_NAME> <BLOCK_NAME> <BLOCK_TYPE> <TARGET_NAME> <CHILD_DATES>

where,

<PREFIX> is BD U|L

<DIRECTORY_NAME> is the hierarchic path to the block directory.

<BLOCKS.NAME> is the name of the block being built

<BLOCK.TYPE> is the type of block (e.g. PROGRAM, FUNCTION_BLOCK, etc).

<TARGET_NAME> is the name of the current build target

<CHILD_DATES> are the dates of the block's dependent block's most recent build time, source modification and interface modification. The dates are those returned by the *Dates* utility - see the output record format.

For the *Build* and *UnBuild* utilities:-

<PREFIX> <DIRECTORY_NAME> <BLOCK_NAME> <BLOCK_TYPE> <TARGET_NAME>

where,

<PREFIX> is B B|U U|L

CONTROLLED DISTRIBUTION COPY
ONLY IF COLOUR STAMPED ON
CONTROL SHEET.

DOCUMENT
REVISION

4

GCT

GCT Build System

EUROTHERM CONTROLS

© Copyright 1994 Eurotherm Controls Ltd



DOCUMENT NUMBER
HP024674C308

SHT.

9

4.1.4 Output Records

For the *Build* and *UnBuild* utilities all output is regarded as comments to be printed on the standard output channel (GCT build window).

For the *Dates* utility:-

Build. The output

```
B [ <date1> [ <date2> ] ]
```

indicates that the block needs building. The optional dates are intended to provide further information that the object store cannot derive itself.

<date1> is the date that the block's source(s) last changed

<date2> is the date that the block's interface last changed

These two dates (which default to `0xFFFFFFFF`) are returned by the object store to future invocations of *Dates* in the input records for parent blocks.

Don't Build. The output

```
D <date1> [ <date2> [ <date3> ] ]
```

indicates that the block doesn't need building.

<date1> is the last build date for this block

<date2> is the date that the block's source(s) last changed (defaults to <date1>)

<date3> is the date that the block's interface last changed (defaults to <date1>)

Error Message. The output

```
E <error message>
```

indicates an error. The message is written to the error window.


Comment Message. The output

```
# <Comment>
```

indicates an informative comment. The message is written to the output window.

The presence of any error output will indicate a failure which will terminate the build process (unless the option *IgnoreErrors* is set).

The format of a date is unsigned numeric — its exact meaning is up to the build system. The object store only uses the numeric value to determine if a block is older (lower numeric value) or younger (higher numeric value). The child date passed into the *Dates* program is the date given for the child block most recently (highest value) built. This value is derived from the results of calling *Dates* for the child blocks. A 'B' output record without a date is interpreted as the maximum date ($2^{32} - 1$ or `0xFFFFFFFF`) — i.e. in the future.

CONTROLLED DISTRIBUTION COPY ONLY IF COLOUR STAMPED ON CONTROL SHEET.	DOCUMENT REVISION 4	GCT	
		GCT Build System	
EUROTHERM CONTROLS © Copyright 1994 Eurotherm Controls Ltd		DOCUMENT NUMBER	SHT.
		HP024674C308	10

5 Parameterisation

It is intended that in general a library can be built for any target. Target specific parameters for a library are held in the library's library.odt option database section. They are set up as a result of a user action (Files\Build SetUp...) which invokes a target configuration dialog. This dialog is either (i) a target specific utility invoked by the object store, or (ii) a general dialog customised by a list of arguments in the options database.

5.1 The "Standard" Dialog

Note: The features of this section are NOT implemented yet

Most targets will require a few parameters to be held against a library for build purposes. These parameters are held in the library's option database section and can be set up by a standard dialog. The dialog is customised by a list of fields held in the target's option database section (Target:BuildSetUpDialog). The format of these field specifiers is (something like):-

[Parameter] : [Prompt] : [Type] : [Default Value]

where,

- [Parameter] is the name of the parameter to be stored in the library section.
- [Prompt] is the prompt string to be used in the dialog.
- [Type] is, for example, one of

STRING[length]

INT low..high


DIR — for a directory name

CHOICE 'string1' 'string2' 'string3' ...

etc.

6 OptionsDatabase

The build system uses the following options from the options database in addition to those described above:-

CONTROLLED DISTRIBUTION COPY ONLY IF COLOUR STAMPED ON CONTROL SHEET.	DOCUMENT REVISION 4	GCT	
		GCT Build System	
EUROTHERM CONTROLS © Copyright 1994 Eurotherm Controls Ltd		DOCUMENT NUMBER	
		HP024674C308	SHT. 11

Parameter	Facility	Default	Usage	Normal Section
LibraryRootDir		library	Directory under GC-TROOT which holds all libraries for compatability with old versions of FBB which used share	default
ProjectRootDir		project	Directory under GC-TROOT which holds all projects for compatability with old versions of FBB which used conf	default
TargetsSupported			The names of all the targets supported by the installation. This list is offered to the user when he selects Files\Build SetUp... from the main GCT menu.	default
BuildSetUpCommand	Target		The full path name of the command to be executed to set up the build system for a given library (alternative to BuildSetUpDialog)	target
BuildSetUpDialog	Target		List of dialog parameters for a standard build set up for a given library (alternative to BuildSetUpCommand) <i>Not Yet Implemented</i>	target
Target			The name of a target that is currently set up as the target for the library. This gets changed by Files\Build SetUp....	library
BuildCommand	Target	build	The full path name of the command to be executed to perform the BUILD function.	target
UnBuildCommand	Target	unbuild	The full path name of the command to be executed to perform the UNBUILD function.	target
DatesCommand	Target	dates	The full path name of the command to be executed to perform the DATES function.	target

→ ○ ←

CONTROLLED DISTRIBUTION COPY
ONLY IF COLOUR STAMPED ON
CONTROL SHEET.

DOCUMENT
REVISION

4

GCT

GCT Build System

EUROTHERM CONTROLS

© Copyright 1994 Eurotherm Controls Ltd



DOCUMENT NUMBER
HP024674C308

SHT.

12