QMAKE

V4

Make Program

for the GST/Quanta Assembler & Linker

Distributed by Jochen Merz Software

Introduction

QMAKE allows you to make the handling of the Assembler and Linker for assembler programs much more easier and more automatic. It combines all the things of which we thought they are necessary during 8 years of assembly language development. QMAKE is extremely helpful for the development of modular programming and large projects.

QMAKE works on an standard LINK control file which is directly suitable for the GST/Quanta Linker. Additional options are possible, but if they are used in the link control file then the link file cannot be processed directly by the Linker anymore, it has to be processed through QMAKE.

Not all of the features work with the original GST Macro Assembler and Linker, and we recommend that you use it together with the latest Quanta version of both programs. Especially if you encounter strange behaviour of QMAKE together with the assembler/linker, ensure you have the latest versions of QMAC and QLINK. At the time the manual is written, QMAC is Version 1.06 and QLINK is Version 1.03.

QMAKE also supports the Tony Tebby Linker, which is much faster than the GST/Quanta Linker, but does not support all instructions. So beware - if you get the correct result using QLINK and then get a different result using TTs linker, you are probably using one of the features not supported by TTs linker.

Before you start using QMAKE, we suggest you have a look at the configuration options. Without adjusting QMAKE to your system (at least specify the linker you're using) it may not produce the result you expect.

Linker Control file

The standard linker control file (_link) instructions are passed to the linker, with the following exceptions:

INPUT filename_REL
read in a file ending in _REL and include all modules into the link. QMAKE looks for the same filename, ending in _ASM, and compares the date of both files. If the _ASM file is younger, then it is automatically assembled before it is linked. Instead of a _REL file a _LIB file can also be used for INPUT, which puts all modules into the result. With Force Assembly activated, INPUT _LIB's are also force assembled.

INPUTN filename_REL
read in a file ending in _REL and include all modules into the link. QMAKE does not check for a corresponding _ASM file, it simply passes the _REL file without any checks to the linker. This is very useful if you've got some _REL files without having any _ASM files for them.

LIBRARY filename_LIB
searches the filename ending in _LIB for all unresolved references in the link and, if found, includes any module which is required. How to construct _LIB files is also explained in this manual.

Instead of passing a full library, only a single _REL file can be specified. It is checked and treated exactly like INPUT would do, but the linker uses the modules only if required (i.e. XREFed by another module).

Any filename may contain one ore more substitution strings, which are expressed by a hash followed by a number. This allows you to create various versions with the same template control file, for example versions in different languages. QMAKE allows you to use up to 10 different substitution strings, numbered from 0 to 9. The first three ones offer most options, and 3 to 9 can be set during make time (it is very unlikely that you might need more than two anyway).

Summary of instructions

Instr.	Extens.	Action	Result for Linker
input	_rel	search for _asm and if not found	input _rel
		if _rel older than _asm: assemble	input _rel
inputn	_rel	do nothing	input _rel
input	_lib	search for _cctx and _cct and if not found	input _lib
		_cctx found: _lib older than _cctx: append new _lib	input _lib
		_cct found: all _rel files older than _asm: assemble ...	
		... if any assembly happened: append new _lib file	input _lib
library	_rel	if _rel older than _asm: assemble	library _rel
library	_lib	search for _cctx and _cct and if both not found	library _lib
		_cctx found: _lib older than _cctx: append new _lib	library _lib
		_cct found : all files _rel older _asm => assemble	
		_cct found: all _rel files older than _asm: assemble ...	
		... if any assembly happened: append new _lib file	library _lib

Here is an example of a fairly complex control file, called win1_QD_LINK:

```
data 5k                                 some dataspace - it is executable
common dummy                            required for the window definitions
section base                            here is the order of the various sections
section version
section language
section sprite
section program
section utility
program win1_qd_#0                      this is the output file name, language substituted
input win1_qd_head_rel                  various files which have to be used
input win1_qd_parsstr_rel
input win1_qd_version_rel
input win1_qd_sprites_rel
input win1_qd_setup_rel
input win1_qd_conf_prepost_rel
input win1_qd_usemenu_rel
input win1_qd_util_lib                  use all of util_lib
library win1_util_strg_lib              but only required modules from strg_lib
input win1_qd_#0_rel                    here is the language-specific text part
input win1_qd_conf_gen_#0_rel           and various language-specific config files
input win1_qd_conf_files_#0_rel
input win1_qd_conf_edit_#0_rel
input win1_qd_wdef_lib                  we need complete window definitions
input win1_qd_init_rel
input win1_qd_initthg_rel
input win1_qd_act_lib                   and also all action routines
library win1_util_pars_lib              but only parts of the following libraries
library win1_wut_lib
library win1_util_sprite_lib
library win1_util_menus_lib
library win1_uti_lib                    uti_lib first time
input win1_qd_thgif_rel                 again, everything is required
input win1_qd_edit_lib
input win1_qd_qtypchk_rel
input win1_qd_interface_rel
library win1_util_gut_lib               gut contains some useful routines
input win1_qd_line_numbers_rel          but the line numbering has to be complete
library win1_util_cv_lib                finally some useful libraries
library win1_lang_#0_lib
library win1_uti_lib                    there might be more in here
```

When this file is parsed by QMAKE, it generates a linker control file which will be recognised by the standard linker. The file will be called filename win1_QD_TMP_LINK which looks like this:

```
data 5k
common dummy
section base
section version
section language
section sprite
section program
section utility
input win1_qd_head_rel
input win1_qd_parsstr_rel
input win1_qd_version_rel
input win1_qd_sprites_rel
input win1_qd_setup_rel
input win1_qd_conf_prepost_rel
input win1_qd_usemenu_rel
input win1_qd_util_lib
library win1_util_strg_lib
input win1_qd_English_rel
input win1_qd_conf_gen_English_rel
input win1_qd_conf_files_English_rel
input win1_qd_conf_edit_English_rel
input win1_qd_wdef_lib
input win1_qd_init_rel
input win1_qd_initthg_rel
input win1_qd_act_lib
library win1_util_pars_lib
library win1_wut_lib
library win1_util_sprite_lib
library win1_util_menus_lib
library win1_uti_lib
input win1_qd_thgif_rel
input win1_qd_edit_lib
input win1_qd_qtypchk_rel
input win1_qd_interface_rel
library win1_util_gut_lib
input win1_qd_line_numbers_rel
library win1_util_cv_lib
library win1_lang_English_lib
library win1_uti_lib
```

Notice that all the #0's are replace by "English" - giving the right file name.
In the example above, win1_uti_lib is given as a library twice. This is, as the code size is fairly critical and therefore the order of the sections is important, part of the library routines are put into the centre of the file and the rest of it, which is referenced from code after the first "library", comes at the very end.

You probably know how to create a linker control file yourself, and now let's do some real work with the program.

First you have to configure it so that it knows the right names used on your system. Execute the MenuConfig program and select QMAKE so that you can configure it. QMAKE's configuration possibilities are divided into three blocks: General, window and substitution. Here is a list of the current configuration options in General:

Linker type
Leave it set to "GST/Quanta", unless you use Tony Tebby's much faster QJUMP linker, which is not available yet. Please note that not all linker directives are supported by TTs linker.

Assembler control
Defines the assembler control which is passed to the assembler, in case files have to be assembled. You know what the options "Errors", "Nolist" and "List" mean - if not, refer to the assembler manual.

Linker control
Here you've got "CRF" for cross reference, "MAP" for ordinary map or "nothing" for nothing.

Nowinds
Can be used on the improved Quanta version of the assembler and linker to turn off any visible output.

Keep Warnings
Usually, files containing no errors but warnings are deleted. If this option is set to YES, then warnings are kept. In addition, the warnings which warn for $59 and $5A which warn for XREF's are replaced by spaces, giving a better overview over the rest of the files. You are probably more interested in branches which could be short etc. You can choose whether QMAKE should get rid of _ERR or _LIST files which contain warnings but no errors, to keep your harddisk tidied up.

Filter Warnings
Optionally, QMAKE can remove the warnings $59 and $5A (which are XREF out of range warnings) for you to give you a better overview over the remaining warnings. This is sometimes very helpful, especially if you have many XREF.S from window definitions, for example.

Name of Assembler-Thing
If files have to be assembled, QMAKE first looks for an assembler Thing, and if not found, then looks for an assembler executable file. You can delete the name here to disable the use of an assembler Thing.

Name of Linker-Thing
As above, but for the linker.

Assembler Name
Here is the filename of the assembler - which is used if the Thing is not defined or not found.

GST Linker Name
As above, but for the GST/Quanta linker.

QJUMP Linker Name
As above, but for the TT QJUMP linker.

File Extension for QMake Command file
This is usually LINK, but it can be set to any other name (but there's no reason why).

Make-Index Name
Enter here the name of the make_index_obj filename, which creates an index file for QD's HyperHelp. If you haven't got at least QD Version 5, ignore it. The \X option in QMAKEs command string will execute the program specified here after the make run.

Beep after successful operation
As the name says, successful operations are reported to you with a BEEP.

Assembler source file extension
Usually, assembler source filenames end in _ASM. If you wish to change this, then you can do it here.

Relocatable object file extension
Usually, relocatable filenames end in _REL. If you wish to change this, then you can do it here.

More than one QMAKE
Specifies if you want to run more than one QMAKE at the same time. If you want to run more than one QMAKE at the same time, QMAKE adds a job ID to the temporary link file name.

Next, the Window config block:

Initial size
Defines if the window appears in the maximum or minimum possible size (not very useful on 512x256 pixel displays, but very useful on higher resolutions).

Size
Here you specifiy the initial size of the window.

Origin
This and the following two items deal with the window origin.

Colourways
Choose your favourite colours.

The third configuration block allows you to define three default strings for the three first substitution strings.

Execution

Now let us start doing something with QMAKE. When QMAKE is executed, it turns into a button. It assumes a link control file in the current directory, specified by the DATA default. You will see the last part of the directory name in the button. When you DO on the button (i.e. press the right mouse button or ENTER or RETURN), QMAKE grows up to its full size. If you hit the Button (press the left mouse button or SPACE) or if you WAKE it (e.g by a WAKE HOTKEY) then QMAKE starts its job.

Let us assume that you see the full window of QMAKE. At the top, you find the standard item for move, resize (DO on this item toggles between minimum size and maximum size), Quit, Wake (starts the action) and sleep (turns QMAKE into button). Below it, any action is usually reported (unless you configured QMAKE, the assembler or the linker not to do it).

In the centre, you see the command file (usually ending in _LINK or whatever you configured). Select this item to choose another command file. The big 'DO' which is the same as a wake, it simply does the job. In the Status menu, you can turn beeping on and off, make an Index file (for HyperHelp in QD Version 5 or 6), or make all three language-versions (English, German and French are default substitutions for string #0, and a make run is done for every language).

All allows you to change all substitution strings. A Hit in the main window cycles through the pre-defined strings of every substitution for #0 to #2, a Do allows you to edit the current setting.

"Nowinds" can be used only if your assembler and linker knows the nowinds option (Quanta's does, GST's old does not). If nowinds is selected, QMAKE turns into a button during make time, showing only what is going on (assembling or linking). If it is not selected, the window stays big and the action is reported within in the QMAKE window. If QMAKE finds an error during assembly or link, a view window appears and QMAKE tries to start listing at the first error (but does not always succeed in doing so). Anyway, you can scroll through the error report to find the error(s). The error window does not appear for errors, just for warnings. When the assembly and link is successful, QMAKE beeps (if told so) and waits. If there was a warning during link (something, which might be ignored most of the time, but could be useful to note sometimes), two exclamation marks are added to the button name. By

the way, the button may be covered by other windows while being active.

At the bottom you can choose different options for the assembler and linker list file in case you're not happy with your configuration. You find three other items above which control the main work of QMAKE. "Link if possible" is self-explanatory. "Inhibit assembly" does not assemble files, even if the source is younger than the relocatible. "Force assembly" does not look at the file dates, it simply assembles all the files specified by INPUT in the link control file. This options should be used if global keys or data INCLUDE files in the assembler source have changed!

We already explained that the decision whether a file has to be assembled or not depends on the file dates of the _ASM and _REL files. If a file is to be INPUTted and the file name does not end in _REL, then QMAKE assumes that it is okay and does not look for a corresponding _ASM file. Otherwise, both file date are compared and if the _ASM file is older, QMAKE knows that the _REL file is in good state. Otherwise, the _ASM file is assembled.

The last menu item left is "Concatenate". It allows you to do a manual concatenation of a library concatenation file. Any library file (which has to end in _LIB) has to have a concatenation control file (which has to end in _CCT or _CCTX). As a library file consists of concatenated _REL files, QMAKE needs to know which files it has to check and to concatenate. Therefore, a _CCT (or _CCTX) is merely a list of _REL files, but the full filename is required (including drive). An example:

```
win1_util_menus_fsel_rel
win1_util_menus_chsl_rel
win1_util_menus_dsel_rel
win1_util_menus_xsel_rel
win1_util_menus_fileerr_rel
win1_util_menus_dorp_rel
win1_util_menus_view_rel
win1_util_menus_yesno_rel
win1_util_menus_retryquit_rel
win1_util_menus_reqstrg_rel
win1_util_menus_rperr_rel
win1_util_menus_button_rel
win1_util_menus_itsl_rel
win1_util_menus_list_rel
win1_util_menus_swreqst_rel
```

As you can see, only full filenames are listed. When you select "Concatenate", you are requested to select a _CCT or _CCTX file. QMAKE checks all _REL file dates with the corresponding _ASM files in the same way it does for the INPUT files, and assembles if required. Finally, if one or more files had to be assembled, all _REL files are concatenated

resulting in a new _LIB file.

If an error is generated during a Concatenation run, it is stopped. You can re-run the concatenation process by DOing the Concatenate item.

The concatenation process is, of course, done automatically every time QMAKE finds a reference to a _LIB file in the linker control file. When QMAKE finds a _LIB reference, it searches for the corresponding _CCT or _CCTX file and, if found, does the same check as described above for the manual procedure, provided, the file date of the _CCT file is younger than the date of the _LIB file.

The difference between _CCT and _CCTX is that a _CCT is always processed and checked, especially if Force assembly is on. Force assembly does not, however, force the re-assembly of files contained in a _CCTX library!

Dependencies

You can give single or a list of files which are checked in addition to the normal checks to decide whether other files have to be assembled or not. Let us say, you include some keys in an assembler source file. Some of the keys can change without the need to re-assemble the file, but if other keys change which contain important values like memory allocation sizes, then this file has to be re-assembled. As QMake compares the dates of the _ASM and _REL files only to see if the file has to be re-assembled, changing the keys-file would not force it to re-assemble. You can now define relationships to other files.

!! file1[,file2,file3...]
will compare the dates of the file(s) given with the date of the file defined in the next line. Example:

```
!! win1_keys_test
INPUT win1_fred_asm
```

This will re-assemble fred_asm if fred_asm itself has changed (i.e. the file is more recent than fred_obj) or if keys_test has changed (i.e. keys_test is more recent than fred_obj).

You can set up a list of files which is checked on all following files until the list is modified or cancelled:

!+ file1[,file2,file3...]
adds files to the list.

!- file1,[file2,file3...]
removes files from the list.

!---
removes all files from the list.

Example:

```
!+ win1_keys_test,win1_keys_demo    check these files too
INPUT win1_fred_asm                 .. when these files are
INPUT win1_joe_asm                  ... checked at make time
!- win1_keys_demo                   dont check this one
INPUT win1_anne_asm                 ... when this is checked
!---                                wipe list
INPUT win1_jack_asm                 normal mode
```

Startup parameters

QMAKE can work automatically by passing it a command string. Current options for the command string are:

\B	make all languages & kill job
\C Command-File	define command file (w/o filename extension)
\E0	turn off beep
\E1	enable beep
\F1	force assembly
\I1	inhibit assembly
\K0	don't keep warnings
\K1	keep warnings
\L0	nolink
\L1	link if possible
\M0 to \M3	main window colourways
\O0	nowinds
\O1	open window
\Q	quiet mode, don't open window
\S0 to \S3	subwindow colourways
\T0	don't filter warnings
\T1	filter warnings
\U0 to \U3	button colourways
\X	execute BASIC program before QMAKE ends make run
\0 String	Default 1 for Substitute #0
\1 String	Default 2 for Substitute #0
\2 String	Default 3 for Substitute #0
\3 String	Default 1 for Substitute #1
\4 String	Default 2 for Substitute #1
\5 String	Default 3 for Substitute #1
\6 String	Default 1 for Substitute #2
\7 String	Default 2 for Substitute #2
\8 String	Default 3 for Substitute #2

The maximum length for substitution strings is 10 characters.

You cannot combine "open window" and "quiet mode"; QMake will not start with such a setting.

Other QMAKE link file commands

Please note that not all link file commands are supported by all linkers!

ABS
forces the assembler to generate absolute code (requires assembler 1.04 or higher)

EX basicprogramm_bas or **_sav**
allows the execution of an SBASIC program during make. This can be used to automatically generate zipped files after link or copy the result to a different directory or device etc.

GST
tells QMAKE to ignore the QMAKE configuration settings and use the GST linker for this link.

INCBIN file
Include binary file (Quanta assembler only)

PC
forces the assembler to generate PC-relative code (requires assembler 1.04 or higher)

QJUMP
tells QMAKE to ignore the QMAKE configuration settings and use the QJUMP linker for this link.

S2RECORD value where value is a decimal number
It then generates file with the filename extension _SREC, containing the result from the link in standard Motorola S2-Record, starting at address value. This command is interpreted by QMAKE only, it is not passed to the Linker.

Other features

Finally, when the make action is in progress, the name found after the PROGRAM directive in the link file (i.e. the result after the LINK) is stuffed into the HOTKEY buffer, ready to be EXECuted, LRESPRed or QMONed. For your convenience this happens twice, at the start of the MAKE progress and at the end (you have have done some other work during the make process).

QMAKE uses RAM8_ for all its temporary files and log files etc. Have a look at RAM8_ if you do not exactly know what is going on. Here you also find a special log file of all files assembled by QMAKE - this can be very useful for maintaining older versions of the same source file.