Changes in C++ TargetRTS

from RoseRT to DevOps Model RealTime

Elena Volkova, Senior Software Engineer
HCL

Revision history

Date	Document Version	Target RTS Version (as defined in RTVersion.h)	Description	Author
2014.May.26	1.0	7.0.00	Initial Release	Elena Volkova
2014.June.03	1.1	7.0.00	Added information about Connexis stack size settings change	Elena Volkova
2018.March.19	2.0	7.0.00	Added information about all recent changes up to DevOps Model RealTime 10.2 2018.09	Elena Volkova
2018.July.12	2.1	7.0.00	Updates for Model RealTime 10.2 2018.27 Added support for 64-bit MinGW GCC 8.1.0	Mattias Mohlin
2018.August.20	2.2	7.0.01	Updates for Model RealTime 10.2 2018.33 Added support for Visual Studio 2017 (15.0) Changed product name in printouts	Mattias Mohlin
2018.September.21	2.3	7.0.02	Updates for Model RealTime 10.3 2018.40 JSON Encoder Dynamic string buffer Added support for newer versions of GCC on Linux Renamed files related to Visual Studio 2017 (15.0	Mattias Mohlin
2018.November.27	2.4	7.0.03	Updates for Model RealTime 10.3 2018.48 Pass data through external ports	Mattias Mohlin

			Allow programmatic send, invoke and reply on a port	
2019.March.26	2.5	7.0.04	Updates for Model RealTime 10.3 2019.15 Doxygen comments and improved code indentation| outline	Mattias Mohlin
2019.April.26	2.6	7.0.05	Updates for Model RealTime 10.3 2019.19 Message queue reporting	Alexander Koptelov
2019.June.5	2.7	7.0.06	Updates for Model RealTime 10.3 2019.23 Show variable values when application is running Take data_size into account in RTDynamicStringOutBuffer::write	Mattias Mohlin
2019.August.28	2.8	7.0.07	Updates for Model RealTime 10.3 2019.35 Allow passing 'NULL' strings in observer commands Generate 'Chain' observer events for internal 'Chain' events Report data passed to an incarnating capsule	Alexander Koptelov
2019.December.17	2.9	7.0.08	Updates for Model RealTime 10.3 2020.03 Synchronize access to the RTDaemon's actorInfo array	Alexander Koptelov
2020.September.29	3.0	7.0.09	Updates for Model RealTime 11.0 2020.39 Override key word for virtual functions Null Pointer Constant Dependency Generation	Puneet Choodamani Mattias Mohlin
2020.November.5	3.1	7.1.00	Updates for Model RealTime 11.0 2020.45 Removal of obsolete files related to RTRUNTIMEBC Removal of C-style casts Removal of Rose RT compatibility macros	Mattias Mohlin
2021.March.22	3.2	7.1.01	Updates for Model RealTime 11.0 2021.16 Include RTMessage.h in	Mattias Mohlin

			RTSymmetricSignal.inl	
2021.June.7	3.3	7.1.02	Updates for Model RealTime 11.1 2021.24 New template function RTObject_class::fromType	Mattias Mohlin
2021.August.13	3.4	7.1.03	Updates for Model RealTime 11.1 2021.34 Optional warnings in RTObject_class::install	Mattias Mohlin
2021.September.20	3.5	7.1.04	Updates for Model RealTime 11.1 2021.40 Capsule factories and dependency injection	Mattias Mohlin
2021.October.18	3.6	7.1.05	Updates for Model RealTime 11.1 2021.46 New compile flag for disabling isReferencedBy check Moving event data when sending events	Mattias Mohlin
2022.May.19	3.7	7.1.06	Updates for Model RealTime 11.1 2022.21 Setting timers with std::chrono library types	Mattias Mohlin
2022.Aug.28	3.8	7.1.07	Updates for Model RealTime 11.2 2022.41 Changes related to how evtState is emitted	Rajnesh
2022.Sep.19	3.9	7.1.07	Updates for Model RealTime 11.2 2022.41 Added support for Visual Studio 2022	Rajnesh
2022.Nov.04	4.0	7.1.08	Updates for Model RealTime 11.2 2022.48 Added support for MinGW12.2.0 Fix memory leak in evtMessageQueue	Rajnesh
2023.Jan.23	4.1	7.1.09	Updates for Model RealTime 11.2 2023.04 Moving data when sending event on replicated port	Rajnesh Mattias Mohlin

			Added support for GCC 12.x New debugger command sendToApp Handling messages received before capsule initialization	
2023.March.09	4.2	7.1.10	Updates for Model RealTime 11.3 2023.13 Remove using namespace std	Rajnesh
2023.April.25	4.3	7.1.11	Updates for Model RealTime 11.3 2023.19 Added move constructor for RTTypedValue_RTString Added support for Clang 16.x on Windows Added support for Clang 15.x on VxWorks	Eshita Rajnesh
2023.Aug.29	4.4	7.1.12	Updates for Model RealTime 11.3 2023.35 Compiling the TargetRTS with a compiler not supporting C++ 11	Rajnesh
2023.Oct.20	4.5	7.1.13	Updates for Model RealTime 11.3 2023.42 Add support for long long and unsigned long long primitive Added support for Clang 14.x on MacOS	Rajnesh
2023.Dec.1	4.6	7.1.14 8.0.0	Updates for Model RealTime 11.3 2023.49 and Model RealTime 12.0 Improve Dependency Injection for Optional Parts Version 8.0.0 and Model RealTime 12.0	Mattias Mohlin
2024.Feb.12	5.0 (final)	8.0.1 and later	**NOTE:** For changes in version 8.0.1 and later, refer to the documentation at https://secure-dev-ops.github.io/code-realtime/target-rts/changelog/.	Mattias Mohlin

This document describes all changes made in the C++ TargetRTS for Model RealTime. All changes compared to the version that is included in RoseRT are covered. The purpose of the document is to help users incorporate TargetRTS changes into their own versions of the TargetRTS.

Table of contents

Support invoke() without reply()

Previously the TargetRTS has required that when you invoke() a signal, then the receiver needs to do a reply(). When the callee does not have any data to pass back to the caller, it is unnecessary to require an explicit reply(). Sometimes the caller just wants to ensure that a signal is synchronously sent without getting any return value. It's analogous to calling a void function in C++.

The TargetRTS has therefore been extended so that it now is optional to provide a reply buffer when calling invoke().

These files were updated:

```
<TargetRTS>/include/RTOutSignal.h
<TargetRTS>/include/RTOutSignal.inl
<TargetRTS>/include/RTProtocol.h
<TargetRTS>/src/RTProtocol/invoke.cc
<TargetRTS>/src/RTProtocol/invokeAt.cc
<TargetRTS>/src/RTProtocol/invokeOne.cc
<TargetRTS>/include/RTSymmetricSignal.h
<TargetRTS>/include/RTSymmetricSignal.inl
```

Defer invoked replies

A new function RTActor::sendCopyToMe() has been added to allow a capsule to send a copy of a message to itself.

Example:

```
RTMessage reply;
myport.myevent().invoke(&reply);
sendCopyToMe(&reply);
```

More generally this function can be used whenever a message cannot be fully handled in a single transition.

These files were updated:

```
<TargetRTS>/include/RTActor.h
<TargetRTS>/include/RTActor.inl
<TargetRTS>/include/RTController.h
<TargetRTS>/include/RTController.inl
```

Remove usage of obsolete RTRUNTIMEBC macro

Removed conditional sections #if RTRUNTIMEBC ... #endif from the following files:

```
<TargetRTS>/include/RTConfig.h
<TargetRTS>/include/RTMessage.h
<TargetRTS>/include/RTMessage.inl
<TargetRTS>/include/RTProtocol.h
<TargetRTS>/include/RTProtocol.inl
<TargetRTS>/include/RTProtocolDescriptor.h
<TargetRTS>/src/MANIFEST.cpp
<TargetRTS>/src/RTActor/control.cc
<TargetRTS>/src/RTException/descriptor.cc
<TargetRTS>/src/RTExternal/descriptor.cc
<TargetRTS>/src/RTFrame/descriptor.cc
<TargetRTS>/src/RTLog/descriptor.cc
```

```
<TargetRTS>/src/RTRootProtocol/descriptor.cc
<TargetRTS>/src/RTTiming/descriptor.cc
<TargetRTS>/target/SLED10T/RTTarget.h
<TargetRTS>/target/SLED10x64T/RTTarget.h
<TargetRTS>/target/VxWorks60T/RTTarget.h
<TargetRTS>/target/VxWorks64RTPT/rttarget.h
```

Remove obsolete target configurations

Removed the following folders from <TargetRTS>/config/ :

```
AIX4S.ppc-gnu-2.8.1
AIX4T.ppc-gnu-2.8.1
CHORUS40T.ppc603-egcs-2.91.66
IRIX6S.r4400-gnu-2.8.1
IRIX6T.r4400-gnu-2.8.1
LYNX30S.ppc-cygnus-2.7-97r1
LYNX30S.x86-cygnus-2.7-97r1
LYNX30T.ppc-cygnus-2.7-97r1
LYNX30T.x86-cygnus-2.7-97r1
LYNX31S.ppc-gnupro-2.9-98r2
LYNX31T.ppc-gnupro-2.9-98r2
NT40CygwinT.x86-cygwin-gnu-2.95.3-5
NT40CygwinT.x86-cygwin-gnu-3.2
NT40CygwinT.x86-cygwin-gnu-3.4.4-3
NT40CygwinT.x86-cygwin-gnu-4.5.3
NUCLEUS11T.ppc860-Diab-4.2b
NUCLEUS11T.x86-VisualC++-6.0
PSOS25T.ppc603-Diab-4.2b
PSOS25T.ppc860-Diab-4.2b
QNX4S.x86-WC++-10.6
REDHAT61S.x86-egcs-2.91.66
REDHAT61T.x86-egcs-2.91.66
REDHAT73MT.x86-gnu-3.2
REDHAT73T.x86-gnu-3.2
REDHAT80MT.x86-gnu-3.2
REDHAT80T.x86-gnu-3.2
SOLARIS10SPARC64T.sparc64-gnu-3.4.6
SOLARIS10T.sparc-gnu-3.4.6
SUN5MT.sparc-SunC++-5.0
SUN5S.sparc-SunC++-5.0
SUN5S.sparc-gnu-2.7.2.3
SUN5S.sparc-gnu-2.8.1
SUN5S.sparc-gnu-2.95.1
SUN5T.sparc-SunC++-5.0
SUN5T.sparc-SunC++-5.3
SUN5T.sparc-gnu-2.7.2.3
SUN5T.sparc-gnu-2.8.1
SUN5T.sparc-gnu-2.95.1
SUN5T.sparc-gnu-3.0.4
TORNADO101T.i960-cygnus-2.7.2-960126
TORNADO101T.m68040-cygnus-2.7.2-960126
TORNADO101T.ppc-cygnus-2.7.2-960126
TORNADO101T.ppc860-cygnus-2.7.2-960126
TORNADO101T.x86-cygnus-2.7.2-960126
TORNADO22T.ppc603-Diab-5.0.1
TORNADO22T.ppc603-gnu-2.96
TORNADO22T.simnt-gnu-2.96
TORNADO22T.simsparcsolaris-gnu-2.96
TORNADO2T.m68040-cygnus-2.7.2-960126
TORNADO2T.ppc-cygnus-2.7.2-960126
TORNADO2T.ppc603-GreenVX-1.8.9
TORNADO2T.ppc603-GreenVX-2.0
TORNADO2T.ppc860-cygnus-2.7.2-960126
```

```
TORNADO2T.simpc-egcs-2.90.29
TORNADO2T.simso-cygnus-2.7.2-960126
TORNADO2T.x86-cygnus-2.7.2-960126
UNIXWARE701S.x86-SDK-3.0
UNIXWARE701T.x86-SDK-3.0
VRTX4T.ppc603-Microtec-1.3C
VRTX4T.ppc603-Microtec-1.4
VxWorksAE10T.ppc603-gnu-2.96
VxWorksAE10T.simnt-gnu-2.96
VxWorksAE10T.simsparcsolaris-gnu-2.96
WINCE300T.arm-eMVisualC++-3.0
WINCE300T.mips-eMVisualC++-3.0
WINCE300T.sh3-eMVisualC++-3.0
```

Removed the following folders from <TargetRTS>/src/target/ :

```
AIX4
CLASSIX31
IRIX6
LYNX2
NT40Cygwin
NUCLEUS
PSOS2
QNX4
SOLARIS10
SOLARIS10SPARC64
SUN5
TORNADO1
UNIXWARE212
VRTX3
WINCE
```

Removed the following folders from <TargetRTS>/target/ :

```
AIX4S
AIX4T
CHORUS40T
IRIX6S
IRIX6T
LYNX30S
LYNX30T
LYNX31S
LYNX31T
NT40CygwinT
NUCLEUS11T
PSOS25T
QNX4S
REDHAT61S
REDHAT61T
REDHAT73T
REDHAT80T
SOLARIS10SPARC64T
SOLARIS10T
SUN5MT
SUN5S
SUN5T
TORNADO101T
TORNADO22T
TORNADO2T
UNIXWARE701S
UNIXWARE701T
```

```
VRTX4T
VxWorksAE10T
WINCE300T
```

Removed the following folders from <TargetRTS>/lib/ :

```
CHORUS40T.ppc603-egcs-2.91.66
LYNX30T.ppc-cygnus-2.7-97r1
LYNX30T.x86-cygnus-2.7-97r1
LYNX31T.ppc-gnupro-2.9-98r2
NT40CygwinT.x86-cygwin-gnu-3.2
NT40CygwinT.x86-cygwin-gnu-3.4.4-3
NT40CygwinT.x86-cygwin-gnu-4.5.3
NT40MT.x86-VisualC++-6.0
REDHAT73MT.x86-gnu-3.2
REDHAT73T.x86-gnu-3.2
REDHAT80MT.x86-gnu-3.2
REDHAT80T.x86-gnu-3.2
SOLARIS10SPARC64T.sparc64-gnu-3.4.6
SOLARIS10T.sparc-gnu-3.4.6
SUN5MT.sparc-SunC++-5.0
SUN5T.sparc-SunC++-5.0
SUN5T.sparc-SunC++-5.3
SUN5T.sparc-gnu-2.8.1
SUN5T.sparc-gnu-2.95.1
SUN5T.sparc-gnu-3.0.4
TORNADO22T.ppc603-gnu-2.96
TORNADO22T.simsparcsolaris-gnu-2.96
TORNADO2T.m68040-cygnus-2.7.2-960126
TORNADO2T.ppc-cygnus-2.7.2-960126
TORNADO2T.ppc860-cygnus-2.7.2-960126
TORNADO2T.x86-cygnus-2.7.2-960126
WINCE300T.sh3-eMVisualC++-3.0
```

Added support for 64-bit Visual Studio and some other new target configurations

The TargetRTS now contain config and libset files for all cases of 64-bit compilation: native (when building on 64-bit platform) and cross platform (when building on 32-bit platform but targeting a 64-bit platform).

Added the following folders:

```
+ <TargetRTS>/config/NT40T.x64-VisualC++-10.0/
+ <TargetRTS>/config/NT40T.x64-VisualC++-9.0/
+ <TargetRTS>/config/NT40T.x86_x64_cross-VisualC++-10.0/
+ <TargetRTS>/config/NT40T.x86_x64_cross-VisualC++-9.0/
+ <TargetRTS>/config/NT40T.x86-VisualC++-11.0/
+ <TargetRTS>/config/NT40T.x86-VisualC++-10.0/
+ <TargetRTS>/config/NT40T.x86-VisualC++-9.0/
...
```

Updated Copyright texts

Updated a comment holding copyright note in all files.

Fix problem in main.mk to make it work for MinGW

Fixed make rules for $(ACTOR_LIB) and $(DATA_LIB). Replaced "%A.olist" with $(A_OBJECTS) and "%D.olist" with $(D_OBJECTS)

This file was updated:

```
<TargetRTS>/src/main.mk
```

Update stack size settings for Connexis on SLED10x64T

During testing we found a crash in gethostbyname() function call on Linux (SLED10x64T) for models using Connexis. The problem is in the stack size for Connexis threads.

gethostbyname() allocates its return value on the stack and if there is no more space, the function crashes.

Default stack sizes are set up in <TargetRTS>/target/SLED10x64T/RTDcsTarget.h:

```
// Stack size of Connexis Transport thread
#define CNXTTP_STACK 8000
// Stack size of Connexis Helper thread
#define CNXTHTP_STACK 16000
// Recommended minimum stack size for Connexis DCS and Locator Capsules
#define CNXDTP_STACK 8000
// Stack size of Connexis debug Agent thread
#define CNXATP_STACK 8000
// Stack size of Connexis ORB threads
#define CNXODP_STACK 10000
```

We have increased the default stack sizes for SLED 64.

Better error reporting in Perl scripts

These files were updated:

```
<TargetRTS>/codegen/rtcomp.pl
<TargetRTS>/codegen/rterror.pl
```

Added support for Quantify and Pure Coverage in rtlink.pl

These files were updated:

```
<TargetRTS>/codegen/rtlink.pl
```

Improved configuration of paths for GNU compiler in setup.pl

These files were updated:

```
<TargetRTS>/config/NoRTOSS.sparc-gnu-2.8.1/setup.pl
<TargetRTS>/config/NoRTOSS.x86-gnu-3.2/setup.pl
```

Added class-scoped state entry operation rtgStateEntry

Support has been added for a class-scoped state entry operation for capsules and passive classes with state machines.

For capsules, a virtual operation rtgStateEntry has been added to the base class for all capsules, RTActor:

```
class RTActor
{ ... protected: virtual void rtgStateEntry( void ); ... }
```

This operation has a default implementation in the RTActor class:

```
void RTActor::rtgStateEntry( void )
{
// by default class-scoped state entry operation does nothing
}
```

This operation is called in RTActor::enterState(int s) right before the local state-entry operation enterStateV():

```
setCurrentState( s );
rtgStateEntry(); //class-scoped state entry operation
enterStateV();
```

To define a class-scoped state-entry operation for a capsule, you need to add an operation

```
void rtgStateEntry()
```

to it. It’s important to mark it as Virtual so it will be generated as a virtual function (otherwise it will not work as intended). This operation can be inherited or redefined as a normal virtual operation. If a capsule does not define rtgStateEntry, the default implementation from RTActor will be called and it will do nothing.

A typical usage of this feature could be to trace the state name whenever a state is entered.

For passive classes with state machines the support for a class-scoped state entry operation is handled by the code generator. It checks if the passive class contains an operation named rtgStateEntry, local or inherited, and includes the call to it into the generated code.

These files were updated:

```
<TargetRTS>/include/RTActor.h
<TargetRTS>/src/RTActor/enterState.cc
<TargetRTS>/src/RTActor/enterStateV.cc
```

Passing information about initiator to some RTActor functions

This change was made for better error reporting, by setting errors to the correct context.

These files were updated:

```
<TargetRTS>/include/RTActor.h
<TargetRTS>/include/RTActorRef.h
<TargetRTS>/include/RTRelayPort.h
<TargetRTS>/src/RTActor/removeAllImports.cc
<TargetRTS>/src/RTActor/removeImport.cc
<TargetRTS>/src/RTActor/unbind.cc
<TargetRTS>/src/RTActorRef/bind_ports.cc
<TargetRTS>/src/RTActorRef/deport_one.cc
<TargetRTS>/src/RTActorRef/deport_RTActor.cc
<TargetRTS>/src/RTActorRef/destroy_RTActor.cc
<TargetRTS>/src/RTRelayPort/unreserve.cc
```

Added support for unsigned integer basic data types to ascii encoding/decoding and logging functions

These files were updated:

```
<TargetRTS>/include/RTAsciiDecoding.h
<TargetRTS>/include/RTAsciiEncoding.h
<TargetRTS>/include/RTDecoding.h
<TargetRTS>/include/RTEncoding.h
<TargetRTS>/include/RTLog.h
<TargetRTS>/include/RTObject_class.h
<TargetRTS>/src/MANIFEST.cpp
```

These files were added:

```
+ <TargetRTS>/src/RTAsciiDecoding/get_unsignedchar.cc
+ <TargetRTS>/src/RTAsciiDecoding/get_unsignedint.cc
+ <TargetRTS>/src/RTAsciiDecoding/get_unsignedlong.cc
+ <TargetRTS>/src/RTAsciiDecoding/get_unsignedshort.cc
+ <TargetRTS>/src/RTAsciiEncoding/put_unsignedchar.cc
+ <TargetRTS>/src/RTAsciiEncoding/put_unsignedint.cc
+ <TargetRTS>/src/RTAsciiEncoding/put_unsignedlong.cc
+ <TargetRTS>/src/RTAsciiEncoding/put_unsignedshort.cc
+ <TargetRTS>/src/RTLog/show_unsignedshort.cc
+ <TargetRTS>/src/RTLog/show_unsignedlong.cc
+ <TargetRTS>/src/RTLog/show_unsignedint.cc
+ <TargetRTS>/src/RTLog/show_unsignedchar.cc
+ <TargetRTS>/src/RTLog/log_unsignedshort.cc
+ <TargetRTS>/src/RTLog/log_unsignedlong.cc
+ <TargetRTS>/src/RTLog/log_unsignedint.cc
+ <TargetRTS>/src/RTLog/log_unsignedchar.cc
+ <TargetRTS>/src/RTObject_class/unsignedshort.cc
+ <TargetRTS>/src/RTObject_class/unsignedlong.cc
+ <TargetRTS>/src/RTObject_class/unsignedint.cc
+ <TargetRTS>/src/RTObject_class/unsignedchar.cc
```

Added more accurate error message about no free ports triggered by FrameImport

A new, more specific, error message has been added and may be reported during frame import. Now the following message will be printed: “Not enough free ports.” instead of the more general one "Not a compatible subclass.".

These files were updated:

```
<TargetRTS>/include/RTController.h
<TargetRTS>/src/RTActorRef/import.cc
<TargetRTS>/src/RTController/strerror.cc
```

Added support for IPv6 protocol

This file was updated:

```
<TargetRTS>/include/RTTcpSocket.h
```

Print information about port index when receiving unexpected message to port

The port index has been added to the “Unexpected signal” error message.

This file was updated:

```
<TargetRTS>/src/RTActor/badMessage.cc
```

Improved TargetRTS multi-threaded mode (when USE_THREADS is defined)

These files were updated:

```
<TargetRTS>/src/RTController/dt.cc
<TargetRTS>/src/RTPeerController/shutdown.cc
<TargetRTS>/src/RTProtocol/reply.cc
<TargetRTS>/src/RTProtocol/sendOne.cc
```

Added support for 64-bit MinGW GCC 8.1.0

The TargetRTS now contains new config and libset files and prebuilt libraries for supporting 64-bit compilation with gcc 8.1.0 from MinGW.

Added the following folders:

```
+ <TargetRTS>/config/MinGwT.x64-MinGw-gnu-8.1.0/
+ <TargetRTS>/libset/x64-MinGw-gnu-8.1.0/
+ <TargetRTS>/lib/MinGwT.x64-MinGw-gnu-8.1.0/
```

Added support for Visual Studio 2017 (15.0)

The TargetRTS now contains new config and libset files and prebuilt libraries for supporting compilation with Visual Studio 2017 (ver 15.0), for both 32 (x86) and 64 bit (x64).

Added the following folders:

```
+ <TargetRTS>/config/WinT.x64-VisualC++-15.0/
+ <TargetRTS>/config/WinT.x86-VisualC++-15.0/
+ <TargetRTS>/libset/x64-VisualC++-15.0/
+ <TargetRTS>/libset/x86-VisualC++-15.0/
+ <TargetRTS>/lib/WinT.x64-VisualC++-15.0/
+ <TargetRTS>/lib/WinT.x86-VisualC++-15.0/
```

Changed product name in printouts

The string “Model RealTime” in printouts was changed to “RT” and “rsart” was changed to “targetRTS”.

Updated the following files:

```
<TargetRTS>/include/RTVersion.h
<TargetRTS>/src/RTCmdLineObserver/ct.cc
<TargetRTS>/src/RTDebugger/checkObservability.cc
<TargetRTS>/src/RTMain/mainLine.cc
<TargetRTS>/src/RTToolsetObserver/sendTxBuffer.cc
<TargetRTS>/src/RTWebObserver/ct.cc
```

JSON Encoder

A new encoder class is now available for encoding messages and data to JSON. It can be used as an alternative to the default ASCII encoder.

Added the following files:

```
+ <TargetRTS>/include/RTJsonEncoding.h
+ <TargetRTS>/src/RTJsonEncoding/ct.cc
+ <TargetRTS>/src/RTJsonEncoding/dt.cc
+ <TargetRTS>/src/RTJsonEncoding/put.cc
+ <TargetRTS>/src/RTJsonEncoding/putFields.cc
+ <TargetRTS>/src/RTJsonEncoding/put_array.cc
+ <TargetRTS>/src/RTJsonEncoding/put_char.cc
+ <TargetRTS>/src/RTJsonEncoding/put_msg.cc
+ <TargetRTS>/src/RTEncoding/put_string.cc
```

Updated the following files:

```
<TargetRTS>/include/RTAsciiEncoding.h
<TargetRTS>/include/RTEncoding.h
<TargetRTS>/include/RTVersion.h
<TargetRTS>/src/RTManifest.cpp
```

Dynamic string buffer

A new class implementing a dynamic string buffer is now available. It can be used as an alternative to RTMemoryOutBuffer, for example when the size of an encoded message is not known. Its implementation is based on STL's string class.

Added the following files:

```
+ <TargetRTS>/include/RTDynamicStringOutBuffer.h
+ <TargetRTS>/include/RTConfig.h
+ <TargetRTS>/src/RTDynamicStringOutBuffer/ct.cc
+ <TargetRTS>/src/RTDynamicStringOutBuffer/dt.cc
+ <TargetRTS>/src/RTDynamicStringOutBuffer/flush.cc
+ <TargetRTS>/src/RTDynamicStringOutBuffer/getString.cc
+ <TargetRTS>/src/RTDynamicStringOutBuffer/unused.cc
+ <TargetRTS>/src/RTDynamicStringOutBuffer/write.cc
```

Updated the following files:

```
<TargetRTS>/src/RTManifest.cpp
<TargetRTS>/include/RTConfig.h
```

Added support for newer versions of GCC on Linux

The TargetRTS now contains new config and libset files and prebuilt libraries for supporting compilation with newer versions of GCC for 64 bits on Linux (versions 4.8.5 and 7.3.1). Files related to obsolete versions of the TargetRTS were also removed.

Added the following files and folders:

```
+ <TargetRTS>/config/LinuxT.x64-gcc-4.x/config.mk
+ <TargetRTS>/config/LinuxT.x64-gcc-4.x/setup.pl
+ <TargetRTS>/config/LinuxT.x64-gcc-7.x/config.mk
+ <TargetRTS>/config/LinuxT.x64-gcc-7.x/setup.pl
+ <TargetRTS>/lib/LinuxT.x64-gcc-4.x/
+ <TargetRTS>/lib/LinuxT.x64-gcc-7.x/
+ <TargetRTS>/libset/x64-gcc-4.x/libset.mk
```

```
+ <TargetRTS>/libset/x64-gcc-7.x/libset.mk
+ <TargetRTS>/src/target/linux/MAIN/main.cc
+ <TargetRTS>/src/target/linux/RTActorSlot.h
+ <TargetRTS>/src/target/linux/RTActorSlot.inl
+ <TargetRTS>/src/target/linux/RTAsciiDecoding/get_address.cc
+ <TargetRTS>/src/target/linux/RTMutex.h
+ <TargetRTS>/src/target/linux/RTMutex/ct.cc
+ <TargetRTS>/src/target/linux/RTMutex/dt.cc
+ <TargetRTS>/src/target/linux/RTMutex/enter.cc
+ <TargetRTS>/src/target/linux/RTMutex/leave.cc
+ <TargetRTS>/src/target/linux/RTSyncObject.h
+ <TargetRTS>/src/target/linux/RTSyncObject/ct.cc
+ <TargetRTS>/src/target/linux/RTSyncObject/dt.cc
+ <TargetRTS>/src/target/linux/RTSyncObject/signal.cc
+ <TargetRTS>/src/target/linux/RTSyncObject/timedwait.cc
+ <TargetRTS>/src/target/linux/RTSyncObject/wait.cc
+ <TargetRTS>/src/target/linux/RTThread/ct.cc
+ <TargetRTS>/src/target/linux/RTThreadInfo.h
+ <TargetRTS>/src/target/linux/RTTimespec/getclock.cc
+ <TargetRTS>/src/target/linux/RTtcp.h
+ <TargetRTS>/target/LinuxT/RTDcsTarget.h
+ <TargetRTS>/target/LinuxT/RTTarget.h
+ <TargetRTS>/target/LinuxT/target.mk
```

Removed the following folders:

```
- <TargetRTS>/codegen/compiler/Diab/
- <TargetRTS>/codegen/compiler/Gimpel/
- <TargetRTS>/codegen/compiler/Green/
- <TargetRTS>/codegen/compiler/HPC++/
- <TargetRTS>/codegen/compiler/Microtec/
- <TargetRTS>/codegen/compiler/SDK/
- <TargetRTS>/codegen/compiler/SGC++/
- <TargetRTS>/codegen/compiler/SunC++/
- <TargetRTS>/codegen/compiler/WC++/
- <TargetRTS>/codegen/compiler/cset/
- <TargetRTS>/codegen/compiler/cygnus/
- <TargetRTS>/config/NT40MinGwT.x86-MinGw-gnu-3.4.5/
- <TargetRTS>/config/NT40MinGwT.x86-MinGw-gnu-4.7.0/
- <TargetRTS>/config/NT40T.x64-VisualC++-10.0/
- <TargetRTS>/config/NT40T.x64-VisualC++-9.0/
- <TargetRTS>/config/NT40T.x86-VisualC++-11.0/
- <TargetRTS>/config/NT40T.x86-VisualC++-10.0/
- <TargetRTS>/config/NT40T.x86-VisualC++-4.2/
- <TargetRTS>/config/NT40T.x86-VisualC++-5.0/
- <TargetRTS>/config/NT40T.x86-VisualC++-6.0/
- <TargetRTS>/config/NT40T.x86-VisualC++-7.0/
- <TargetRTS>/config/NT40T.x86-VisualC++-9.0/
- <TargetRTS>/config/NT40T.x86_x64_cross-VisualC++-10.0/
- <TargetRTS>/config/NT40T.x86_x64_cross-VisualC++-9.0/
- <TargetRTS>/config/NoRTOSS.sparc-gnu-2.8.1/
- <TargetRTS>/config/NoRTOSS.x86-VisualC++-6.0/
- <TargetRTS>/config/NoRTOSS.x86-gnu-3.2/
- <TargetRTS>/config/OSE411T.ppc603-Diab-4.3f/
- <TargetRTS>/config/OSE411T.sparc-gnu-2.95.1/
- <TargetRTS>/config/VXWorks60T.ppc603-Diab-5.2.2/
- <TargetRTS>/config/VXWorks60T.simnt-Diab-5.2.2/
- <TargetRTS>/config/VxWorks60T.ppc603-gnu-3.3.2/
- <TargetRTS>/config/VxWorks60T.simnt-gnu-3.3.2/
- <TargetRTS>/config/VxWorks64RTPT.ppc603-Diab-5.5.0-RTP/
- <TargetRTS>/config/VxWorks64RTPT.ppc603-gnu-3.4.4-RTP/
- <TargetRTS>/config/VxWorks64RTPT.simnt-Diab-5.5.0-RTP/
- <TargetRTS>/config/VxWorks64RTPT.simnt-gnu-3.4.4-RTP/
```

```
- <TargetRTS>/lib/NT40MinGwT.x86-MinGw-gnu-4.7.0/
- <TargetRTS>/lib/NT40T.x64-VisualC++-9.0/
- <TargetRTS>/lib/NT40T.x86-VisualC++-9.0/
- <TargetRTS>/lib/SLED10x64T.x64-gnu-4.1/
- <TargetRTS>/libset/ppc603-Diab-4.3f/
- <TargetRTS>/libset/ppc603-Diab-5.2.2/
- <TargetRTS>/libset/ppc603-Diab-5.5.0-RTP/
- <TargetRTS>/libset/ppc603-gnu-3.3.2/
- <TargetRTS>/libset/ppc603-gnu-3.4.4-RTP/
- <TargetRTS>/libset/simnt-Diab-5.2.2/
- <TargetRTS>/libset/simnt-Diab-5.5.0-RTP/
- <TargetRTS>/libset/simnt-gnu-3.3.2/
- <TargetRTS>/libset/simnt-gnu-3.4.4-RTP/
- <TargetRTS>/libset/sparc-gnu-2.8.1/
- <TargetRTS>/libset/sparc-gnu-2.95.1/
- <TargetRTS>/libset/x64-VisualC++-10.0/
- <TargetRTS>/libset/x64-VisualC++-9.0/
- <TargetRTS>/libset/x64-gnu-4.1/
- <TargetRTS>/libset/x86-MinGw-gnu-3.4.5/
- <TargetRTS>/libset/x86-MinGw-gnu-4.7.0/
- <TargetRTS>/libset/x86-SDK-3.0/
- <TargetRTS>/libset/x86-VisualC++-10.0/
- <TargetRTS>/libset/x86-VisualC++-11.0/
- <TargetRTS>/libset/x86-VisualC++-4.2/
- <TargetRTS>/libset/x86-VisualC++-5.0/
- <TargetRTS>/libset/x86-VisualC++-6.0/
- <TargetRTS>/libset/x86-VisualC++-7.0/
- <TargetRTS>/libset/x86-VisualC++-9.0/
- <TargetRTS>/libset/x86_x64_cross-VisualC++-10.0/
- <TargetRTS>/libset/x86_x64_cross-VisualC++-9.0/
- <TargetRTS>/src/target/LINUX/
- <TargetRTS>/src/target/NoRTOS/
- <TargetRTS>/src/target/OSE3/
- <TargetRTS>/src/target/SLED10/
- <TargetRTS>/src/target/SLED10x64/
- <TargetRTS>/src/target/VXWorks60D/
- <TargetRTS>/src/target/VxWorks60/
- <TargetRTS>/src/target/VxWorks64RTP/
- <TargetRTS>/target/NoRTOSS/
- <TargetRTS>/target/OSE411T/
- <TargetRTS>/target/SLED10T/
- <TargetRTS>/target/SLED10x64T/
- <TargetRTS>/target/VxWorks60T/
- <TargetRTS>/target/VxWorks64RTPT/
- <TargetRTS>/target/XEC68V3T/
- <TargetRTS>/target/NT40MinGwT/
- <TargetRTS>/target/NT40T/
```

Renamed files related to Visual Studio 2017 (15.0)

Files related to the target configuration for Visual Studio 2017 were renamed to be more consistent with how other target configuration files are named.

Added the following files and folders:

```
+ <TargetRTS>/config/WinT.x64-MinGw-8.1.0/
+ <TargetRTS>/lib/WinT.x64-MinGw-8.1.0/
+ <TargetRTS>/libset/x64-MinGw-8.1.0/
+ <TargetRTS>/src/target/win/
```

Removed the following files and folders:

```
- <TargetRTS>/config/MinGwT.x64-MinGw-gnu-8.1.0/
```

```
- <TargetRTS>/lib/MinGwT.x64-MinGw-gnu-8.1.0/
- <TargetRTS>/libset/x64-MinGw-gnu-8.1.0/
- <TargetRTS>/src/target/NT40MinGw/MAIN/main.cc
- <TargetRTS>/src/target/NT40/
- <TargetRTS>/target/MinGwT/
```

Changed the following files and folders:

```
<TargetRTS>/config/WinT.x64-VisualC++-15.0/config.mk
<TargetRTS>/config/WinT.x86-VisualC++-15.0/config.mk
```

Pass data through external ports

The External::Base::raise() function now accepts a data object, and functions were added for storing and retrieving data objects on the external port itself.

Added the following files and folders:

```
+ <TargetRTS>/src/RTExternal/ExternalData.cc
```

Changed the following files and folders:

```
<TargetRTS>/include/RTExternal.h
<TargetRTS>/include/RTExternal.inl
<TargetRTS>/src/manifest.cpp
<TargetRTS>/src/RTExternal/descriptor.cc
<TargetRTS>/src/RTExternal/raise.cc
```

Allow programmatic send, invoke and reply on a port

The functions in RTProtocol.h for sending events, invoking events and replying on invocation events were made public to make it possible to programmatically perform these operations on a generic port.

Changed the following files:

```
<TargetRTS>/include/RTProtocol.h
```

Doxygen comments and improved code indentation

Doxygen comments have been added in many files, and the code indentation has been improved to make the code more readable.

Changed the following files:

```
<TargetRTS>/include/RTThread.h
<TargetRTS>/include/RTJob.h
<TargetRTS>/src/RTStreamBuffer/write.cc
<TargetRTS>/src/RTMemoryOutBuffer/write.cc
<TargetRTS>/src/RTDiagBuffer/flush.cc
<TargetRTS>/src/RTTcpOutBuffer/flush.cc
<TargetRTS>/include/RTOBuffer.h
<TargetRTS>/include/RTAsciiDecoding.h
<TargetRTS>/include/RTProtocol.h
<TargetRTS>/include/RTAsciiEncoding.h
<TargetRTS>/src/RTAsciiEncoding/put.cc
<TargetRTS>/include/RTDataObject.h
<TargetRTS>/include/RTDynamicStringOutBuffer.h
```

```
<TargetRTS>/include/RTException.h
<TargetRTS>/include/RTMessageQ.h
<TargetRTS>/include/RTSoleController.h
<TargetRTS>/include/RTString.h
<TargetRTS>/include/RTActor_class.h
<TargetRTS>/include/RTController.h
<TargetRTS>/include/RTPeerController.h
<TargetRTS>/src/RTController/abort.cc
<TargetRTS>/src/RTController/dispatch.cc
<TargetRTS>/src/RTController/kill.cc
<TargetRTS>/src/RTController/printStats.cc
<TargetRTS>/src/RTMain/shutdown.cc
<TargetRTS>/include/RTActorRef.h
<TargetRTS>/src/RTActor/defaultClass.cc
<TargetRTS>/src/RTActor/defaultController.cc
<TargetRTS>/src/RTActor/isReferencedBy.cc
<TargetRTS>/src/RTActorClass/isKindOf.cc
<TargetRTS>/src/RTFrame/incarnationsOf.cc
<TargetRTS>/src/RTMessage/defer.cc
<TargetRTS>/src/RTObject_class/isKindOf.cc
<TargetRTS>/include/RTActorClass.h
<TargetRTS>/include/RTInSignal.h
<TargetRTS>/include/RTLog.h
<TargetRTS>/include/RTMessage.h
<TargetRTS>/include/RTObject_class.h
<TargetRTS>/include/RTOutSignal.h
<TargetRTS>/include/RTSymmetricSignal.h
<TargetRTS>/include/RTTypedValue.h
<TargetRTS>/src/RTActorClass/substitutableFor.cc
<TargetRTS>/include/RTActor.h
<TargetRTS>/include/RTActorId.h
<TargetRTS>/include/RTFrame.h
<TargetRTS>/include/RTTimerId.h
<TargetRTS>/include/RTTimespec.h
<TargetRTS>/include/RTTiming.h
<TargetRTS>/include/RTExternal.h

<TargetRTS>/src/RTDaemon/getMessages.cc
<TargetRTS>/src/RTDaemon/updateSettings.cc
<TargetRTS>/src/RTDebugger/evtMsgQueues.cc
<TargetRTS>/src/RTDebugger/getMessages.cc
<TargetRTS>/src/RTObserver/dispatchEvent.cc
<TargetRTS>/src/RTObserver/evtMessageQueue.cc
<TargetRTS>/src/RTToolsetObserver/encoDeco.cc
<TargetRTS>/src/RTToolsetObserver/endeBoth.cc
<TargetRTS>/src/RTToolsetObserver/endeTarg.cc
<TargetRTS>/src/RTToolsetObserver/evtMessageQueue.cc
<TargetRTS>/src/RTToolsetObserver/handleGetMessages.cc
<TargetRTS>/src/include/RTActorProbe.h
<TargetRTS>/src/include/RTDaemon.h
<TargetRTS>/src/include/RTDebugger.h
<TargetRTS>/src/include/RTEventInfo.h
<TargetRTS>/src/include/RTObserver.h
<TargetRTS>/src/include/RTToolsetObserver.h
<TargetRTS>/src/include/endeBoth.h
<TargetRTS>/src/include/endePriv.h
<TargetRTS>/src/include/endeTarg.h
```

Show variable values when application is running

These changes were made in order to support showing the values of variables in the Variables view without having to suspend the application.

Added the following files:

```
<TargetRTS>/src/RTActorProbe/fieldMonitor.cc
<TargetRTS>/src/RTActorProbe/reportFieldValues.cc
<TargetRTS>/src/RTDebugger/fieldMonitor.cc
```

Changed the following files:

```
<TargetRTS>/src/MANIFEST.cpp
<TargetRTS>/src/RTActor/processHistory.cc
<TargetRTS>/src/RTDaemon/delInfo.cc
<TargetRTS>/src/RTMessage/deliver.cc
<TargetRTS>/src/RTToolsetObserver/handleVariable.cc
<TargetRTS>/src/include/RTActorProbe.h
<TargetRTS>/src/include/RTDebugger.h
```

Take data_size into account in RTDynamicStringOutBuffer::write

A bug was fixed in this function. Previously the data_size parameter was ignored, but now it is respected and controls how many characters are written to the buffer.

Changed the following file:

```
<TargetRTS>/src/RTDynamicStringOutBuffer/write.cc
```

Allow passing ‘NULL’ strings in observer commands

Code that matches e.g. ‘Chain’ event for a probe can match every transition if the corresponding field in the probe is null. But it was not possible to configure the probe that way since passing null strings was not supported.

Now it is possible to use **STR -1** to indicate that the string should be **null**.

Changed the following files:

```
<TargetRTS>/src/RTDaemonInfo/setInfoChain.cc
<TargetRTS>/src/RTToolsetObserver/endeBoth.cc
```

Generate ‘Chain’ observer events for internal ‘Chain’ events

Previously ‘Chain’ events that are triggered on different stages of a transition (pre-exit code, pre-transition effect, post-transition effect, post-entry code) and ‘State’ event that is triggered after executing entry code were converted to ‘MessageIn’ event sent to an observer. That made impossible to track these parts of transitions (as ‘MessageIn’ observer event do not have any information on it transition). In order to generate proper event on entering a state (after transition effect execution and before executing state entry code) this was changed so now ‘Chain’ events are reported to an observer.

Changed the following files:

```
<TargetRTS>/src/RTObserver/dispatchEvent.cc
```

Report data passed to an incarnating capsule

An event ‘MessageOutIn’ is reported on sending events between different contexts.

An ‘Actor’ probe now does not match ‘Chain’ events to avoid receiving duplicates on debugger side.

Changed the following files:

```
<TargetRTS>/src/RTController/send.cc
<TargetRTS>/src/RTDaemonInfo/setInfoActor.cc
```

Synchronize access to the RTDaemon’s `actorInfo` array

One of our customers reported a crash that happens when selecting a capsule with a lot of attributes to view their values. Debugging reports access violation caused by accessing memory via invalid pointer obtained from the `actorInfo` array. The array can be accessed from different concurrent threads (in this case from observer thread to add variable monitor and from controller thread to report variable’s value). Thus there is a possibility of race condition.

To resolve this issue access to the `actorInfo` array should be synchronized. Using `RTMutex` would be too restrictive as it should be possible to have simultaneous read access. Now the new `RTRWLock` primitive is used to allow either shared or exclusive access to the array.

Changed the following files:

```
<TargetRTS>/src/include/RTDaemon.h
<TargetRTS>/src/RTDaemon/addInfo.cc
<TargetRTS>/src/RTDaemon/delInfo.cc
<TargetRTS>/src/RTDaemon/eventMatches.cc
<TargetRTS>/src/RTDaemon/monitorField.cc
<TargetRTS>/src/RTDaemon/posInfo.cc
<TargetRTS>/src/RTDaemon/refresh.cc
<TargetRTS>/src/RTDaemon/setInfo.cc
<TargetRTS>/src/MANIFEST.cpp
```

Added the following files:

```
<TargetRTS>/src/target/linux/RTRWLock.h
<TargetRTS>/src/target/linux/RTRWLock/ct.cc
<TargetRTS>/src/target/linux/RTRWLock/dt.cc
<TargetRTS>/src/target/linux/RTRWLock/enter.cc
<TargetRTS>/src/target/linux/RTRWLock/leave.cc
<TargetRTS>/src/target/win/RTRWLock.h
<TargetRTS>/src/target/win/RTRWLock/ct.cc
<TargetRTS>/src/target/win/RTRWLock/dt.cc
<TargetRTS>/src/target/win/RTRWLock/enter.cc
<TargetRTS>/src/target/win/RTRWLock/leave.cc
```

Override keyword for virtual functions

This change is part of the effort of making the TargetRTS code C++ 11 compliant.

Static code checkers like cppcheck generate a lot of warnings for the missing `override` keyword pertaining to virtual functions. These can be seen when using the GCC compiler with the flag `-Wsuggest-override`. To resolve this, the `override` keyword has been added to virtual functions that are overridden. For GCC targets the `-Wsuggest-override` flag is now used, and the `-std=c++11` flag is also set.

Changed the following files:

```
<TargetRTS>/include/target/RTActiveConnector.h
<TargetRTS>/include/target/RTActorId.h
<TargetRTS>/include/target/RTAsciiDecoding.h
<TargetRTS>/include/target/RTAsciiEncoding.h
<TargetRTS>/include/target/RTBoolean.h
<TargetRTS>/include/target/RTByteBlock.h
<TargetRTS>/include/target/RTCharacter.h
<TargetRTS>/include/target/RTConnector.h
<TargetRTS>/include/target/RTController.h
<TargetRTS>/include/target/RTCustomController.h
<TargetRTS>/include/target/RTDynamicStringOutBuffer.h
<TargetRTS>/include/target/RTEnumerated.h
<TargetRTS>/include/target/RTInteger.h
<TargetRTS>/include/target/RTJsonEncoding.h
<TargetRTS>/include/target/RTLayerConnector.h
<TargetRTS>/include/target/RTMemoryInBuffer.h
<TargetRTS>/include/target/RTMemoryOutBuffer.h
<TargetRTS>/include/target/RTPeerController.h
<TargetRTS>/include/target/RTPointer.h
<TargetRTS>/include/target/RTPriority.h
<TargetRTS>/include/target/RTReal.h
<TargetRTS>/include/target/RTSequence.h
<TargetRTS>/include/target/RTSignal.h
<TargetRTS>/include/target/RTStreamBuffer.h
<TargetRTS>/include/target/RTString.h
<TargetRTS>/include/target/RTTime.h
<TargetRTS>/include/target/RTVAsciiDecoding.h
<TargetRTS>/include/target/RTVEnDecoding.h
<TargetRTS>/include/target/RTVersion.h
<TargetRTS>/src/include/RTController/destory.cc
<TargetRTS>/src/include/RTController/getMessages.cc
<TargetRTS>/src/include/RTProtocol/resize.cc
<TargetRTS>/src/include/RTAbortController.h
<TargetRTS>/src/include/RTActorProbe.h
<TargetRTS>/src/include/RTActorRefProbe.h
<TargetRTS>/src/include/RTCmdLineObserver.h
<TargetRTS>/src/include/RTDebugger.h
<TargetRTS>/src/include/RTDiagBuffer.h
<TargetRTS>/src/include/RTLocalConnector.h
<TargetRTS>/src/include/RTLogBuffer.h
<TargetRTS>/src/include/RTLPurgeFilter.h
<TargetRTS>/src/include/RTRecallFilter.h
<TargetRTS>/src/include/RTSuperActor.h
<TargetRTS>/src/include/RTTcpInBuffer.h
<TargetRTS>/src/include/RTTcpOutBuffer.h
<TargetRTS>/src/include/RTTimerActor.h
<TargetRTS>/src/include/RTTimerController.h
<TargetRTS>/src/include/RTToolsetObserver.h
<TargetRTS>/src/include/RTWebObserver.h
<TargetRTS>/libset/x64-gcc-4.x/libset.mk
<TargetRTS>/libset/x64-gcc-7.x/libset.mk
<TargetRTS>/libset/x64-MinGw-8.1.0/libset.mk
```

NullPointer Constant

This change is part of the effort of making the TargetRTS code C++ 11 compliant.

The TargetRTS were using integer 0 to denote a null pointer, often casted to a pointer type using a C-style cast. These 0's have now been replaced with the C++ 11 `nullptr` constant and the unnecessary casts have also been removed. For GCC targets the `-Wzero-as-null-pointer-constant` flag is now used, in order to get warnings for such problems.

Changed the following files:

```
<TargetRTS>/libset/x64-gcc-4.x/libset.mk
<TargetRTS>/libset/x64-gcc-7.x/libset.mk
<TargetRTS>/libset/x64-MinGw-8.1.0/libset.mk
+ 532 more C++ files in the TargetRTS
```

To find the places to update it's recommended to use a regular expression matching code snippets on the form `(TYPE*) 0` and replacing them with `nullptr`. With GCC the `-Wzero-as-null-pointer-constant` flag helps to point out the code that needs to be updated.

Dependency Generation

In order to support incremental compilation of files that have dependencies on external header files, it's necessary to make a few changes in certain TargetRTS files. This feature relies on the compiler's ability to generate dependency files and is currently only supported for GCC compilers, and only for recursive make files.

Changed the following files:

```
<TargetRTS>/libset/default.mk
<TargetRTS>/libset/x64-gcc-4.x/libset.mk
<TargetRTS>/libset/x64-gcc-7.x/libset.mk
<TargetRTS>/libset/x64-MinGw-8.1.0/libset.mk
```

The file `default.mk` defines two new variables:

- DEP_EXT: The file extension (.d) used for dependency files
- DEP_OPTS: The compiler flags (-MD -MP) that should be set so that dependency files are generated when files are compiled

In the file `libset.mk` you must then add the variable $(DEP_OPTS) to LIBSETCCFLAGS. The generated make file will then automatically include generated dependency files, in order to achieve correct incremental compilation whenever a dependent header file is changed.

Removal of obsolete files related to RTRUNTIMEBC

Due to the removal of the RTRUNTIMEBC macro (see `Remove usage of obsolete RTRUNTIMEBC macro`) certain files have become obsolete and have therefore been removed.

Removed the following files:

```
<TargetRTS>/include/RTGlobalSignal.h
<TargetRTS>/include/RTGlobalSignal.inl
<TargetRTS>/src/RTGlobalSignal/dt.cc
```

```
<TargetRTS>/src/RTGlobalSignal/inlines.cc
<TargetRTS>/src/RTGlobalSignal/unknown.cc
<TargetRTS>/src/RTMessage/getGlobalSignal.cc
<TargetRTS>/src/RTMessage/reply.cc
<TargetRTS>/src/RTProtocol/getGlobalSignal.cc
<TargetRTS>/src/RTProtocol/getLocalSignal.cc
<TargetRTS>/src/RTProtocolDescriptor/getUnknownGlobalSignal.cc
<TargetRTS>/src/RTProtocolDescriptor/getUnknownLocalSignal.cc
```

Also removed entries related to `RTGlobalSignal` from the file

```
<TargetRTS>/src/MANIFEST.cpp
```

Removal of C-style casts

This change is part of the effort of making the TargetRTS code C++ 11 compliant.

C-style casts ((TYPE) expr) have been replaced to avoid warnings when compiling the TargetRTS with modern C++ compilers and static code checker tools. For GCC targets the `-Wuseless-cast` and `-Wold-style-cast` flags are now used, in order to get warnings for such problems.

Each C-style cast has either been removed (if the cast was unnecessary, for example in case of "widening" conversions), replaced with static_cast<> (for example in case of "narrowing" conversions), or replaced with reinterpret_cast<> (for example when casting between unrelated pointer types, or between integral and pointer types). In a few cases a static_cast<> was added even when there was no C-style cast, to avoid warnings related to "narrowing" type conversions.

Also, the GCC flag `-fpermissive` has been removed to surface additional problems related to casts. This revealed two specific problems that now have been fixed:

- The function `RTAsciiDecoding::get_address()` incorrectly decoded an address from text to memory on Windows 64 bit platforms. The Linux specific implementation of this function worked, but has now been replaced with a common implementation that works on all platforms.
- The `RTActorSlot` implementation did not take advantage of the additional size of pointers on 64 bit platforms. As a consequence capsule parts with a multiplicity larger than 32767 occupied twice as much memory as necessary. Also here there was a Linux-specific implementation that has now been replaced with a common implementation.

Changed the following files:

```
<TargetRTS>/libset/x64-gcc-4.x/libset.mk
<TargetRTS>/libset/x64-gcc-7.x/libset.mk
<TargetRTS>/libset/x64-MinGw-8.1.0/libset.mk
<TargetRTS>/libset/x64-VisualC++-15.0/libset.mk
<TargetRTS>/libset/x86-VisualC++-15.0/libset.mk
+ 475 more C++ files in the TargetRTS
```

To find the places to update it's recommended to compile the code with the GCC flags `-Wuseless-cast` and `-Wold-style-cast` and fix reported warnings.

Removed the following files:

```
<TargetRTS>/src/target/linux/RTAsciiDecoding/get_address.cc
<TargetRTS>/src/target/linux/RTActorSlot.h
<TargetRTS>/src/target/linux/RTActorSlot.inl
```

Removal of Rose RT compatibility macros

Macros (and typedefs etc) provided for compatibility with old versions of Rose RT (version 6.4 and earlier) have now been removed. These macros were defined in the file `RTCompatibility.h`. If your code still uses these macros it's recommended to update it. If you still need some of these macros, you will have to define them yourself in a common header file.

Changed the following files:

```
<TargetRTS>/include/RTStructures.h
```

Removed the following files:

```
<TargetRTS>/include/RTCompatibility.h
```

Include RTMessage.h in RTSymmetricSignal.inl

The changes made in `Support invoke() without reply()` introduced a problem that could cause a compile error if RTSymmetricSignal.h is used in a context where RTMessage.h is not included. The solution is to include RTMessage.h in RTSymmetricSignal.inl.

Changed the following files:

```
<TargetRTS>/include/RTSymmetricSignal.inl
```

New template function RTObject_class::fromType

Starting from Model RealTime 11.1 2021.24 the model compiler can generate type descriptors with template parameters. This makes it possible to implement a type descriptor generically for a type with template parameters. In such implementations it's often useful to lookup the type descriptor for a type at compile time, for example the element type of a generic container type. To allow this a new template function was added to the RTObject_class struct:

```
template <typename TYPE> static const RTObject_class * fromType();
```

Specializations of this template function exist for all built-in types, while you have to write your own specializations for any other types used in your application for which this is needed.

Also added printout of a warning if the TargetRTS detects that more than one type descriptor is installed with the same name.

Changed the following files:

```
<TargetRTS>/include/RTObject_class.h
<TargetRTS>/include/RTObject_class.inl
<TargetRTS>/src/RTObject_class/install.cc
```

Optional warnings in RTObject_class::install

The printout of a warning if the TargetRTS detects that more than one type descriptor is installed with the same name (see `New template function RTObject_class::fromType`) can now be avoided by use of a new parameter that was added to the RTObject_class::install() function. The function signature is now:

```
void install( bool warnForDuplicates = true ) const;
```

Hence, the default behavior is still to print the warning, which in most cases is useful and desired. However, if there is legacy code that calls this function, for example in order to register a manually written type descriptor for an external type, and that code may install the same types more than once, then the new parameter can be used to avoid printing a warning. Note, however, that it's usually better to make sure each type is only installed once, as subsequent calls for the same type will not have any effect.

Changed the following files:

```
<TargetRTS>/include/RTObject_class.h
<TargetRTS>/src/RTObject_class/install.cc
```

Capsule factories and dependency injection

The TargetRTS now allows to customize how capsule instances are created and destroyed by defining a so called Capsule Factory (see RTActorFactoryInterface.h). Such a factory can either be provided explicitly when incarnating an optional capsule part by means of a new RTFrame::incarnateCustom() function, or it can be associated with a capsule part by means of a new field RTComponentDescriptor::factory. If a factory is provided by both these means at the same time, the former takes precedence.

A new dependency injection service was also added, implemented by the class RTInjector. It allows to register a create function to be used when incarnating a capsule part, and can for example customize how to create the capsule instance (e.g. provide custom capsule constructor arguments), and which thread to run it in. A capsule factory can delegate requests to create a capsule instance to the dependency injection service, which makes it possible to centralize all dependencies of capsules in an application to a central location (for example the top capsule constructor). It's also possible to change registered create functions dynamically, which makes it possible to at runtime replace a capsule instance created in one way with another capsule instance created another way.

Changed the following files:

```
<TargetRTS>/include/RTActorRef.h
<TargetRTS>/include/RTComponentDescriptor.h
<TargetRTS>/include/RTFrame.h
<TargetRTS>/include/RTStructures.h
<TargetRTS>/src/RTActorRef/destroy_RTActor.cc
<TargetRTS>/src/RTActorRef/destroy.cc
<TargetRTS>/src/RTActorRef/incarnate_void.cc
<TargetRTS>/src/RTActorRef/incarnate2.cc
<TargetRTS>/src/RTActorRef/incarnateAll.cc
<TargetRTS>/src/RTController/destroy.cc
<TargetRTS>/src/MANIFEST.cpp
```

Added the following files:

```
<TargetRTS>/include/RTActorFactory.h
<TargetRTS>/include/RTActorFactoryInterface.h
<TargetRTS>/include/RTDefaultActorFactory.h
<TargetRTS>/include/RTInjector.h
<TargetRTS>/src/RTActorFactory/create.cc
<TargetRTS>/src/RTActorFactory/ct.cc
<TargetRTS>/src/RTActorFactory/destroy.cc
```

```
<TargetRTS>/src/RTFrame/incarnateCustom1.cc
<TargetRTS>/src/RTFrame/incarnateCustom2.cc
<TargetRTS>/src/RTInjector/create.cc
<TargetRTS>/src/RTInjector/ct.cc
<TargetRTS>/src/RTInjector/dt.cc
<TargetRTS>/src/RTInjector/getCreateFunction.cc
<TargetRTS>/src/RTInjector/registerCreateFunction.cc
```

New compile flag for disabling isReferencedBy check

When a capsule instance is imported into a plugin capsule part, by default a run-time check is performed to ensure that importing the capsule instance won't cause a cycle in the reference graph (i.e. a capsule instance must never directly or indirectly contain a capsule part reference to itself). The run-time check is implemented by the function `RTActor::isReferencedBy()`. It is now possible to disable this run-time check to improve the performance of importing.

A new compile flag `RTIMPORT_ISREFERENCEDBY_CHECK` has been introduced. By default (in `RTConfig.h`) it is set to 1 which means the run-time check will be performed. Set it to 0 (in `RTLibSet.h` or `RTTarget.h`) to disable the run-time check (`RTActor::isReferencedBy()` will then be removed).

Changed the following files:

```
<TargetRTS>/include/RTActor.h
<TargetRTS>/include/RTConfig.h
<TargetRTS>/src/RTActorRef/import.cc
<TargetRTS>/src/MANIFEST.cc
```

Moving event data when sending events

A new "move" type descriptor function was introduced to allow event data to be moved instead of copied when sent. The TargetRTS will attempt to move the event data if it is passed as an rvalue reference in the send-statement. For example:

```
MyClass mc;
thePort.theEvent(mc).send(); // Send by copy
thePort.theEvent(std::move(mc)).send(); // Send by move
```

If there is no move function defined in the type descriptor, the event data will be copied like before. Type descriptors for all TargetRTS types were updated, but only RTString has a move function defined.

Changed the following files:

```
<TargetRTS>/include/RTMessage.h
<TargetRTS>/include/RTMessage.inl
<TargetRTS>/include/RTObject_class.h
<TargetRTS>/include/RTObject_class.inl
<TargetRTS>/include/RTOutSignal.h
<TargetRTS>/include/RTOutSignal.inl
<TargetRTS>/include/RTProtocol.h
<TargetRTS>/include/RTString.h
<TargetRTS>/include/RTString.inl
<TargetRTS>/src/RTActorId/getClassData.cc
<TargetRTS>/src/RTBoolean/getClassData.cc
<TargetRTS>/src/RTByteBlock/getClassData.cc
```

```
<TargetRTS>/src/RTCharacter/getClassData.cc
<TargetRTS>/src/RTController/send.cc
<TargetRTS>/src/RTDataObject/getClassData.cc
<TargetRTS>/src/RTDataObject/pRTDataObject.cc
<TargetRTS>/src/RTEnumerated/classData.cc
<TargetRTS>/src/RTInteger/getClassData.cc
<TargetRTS>/src/RTInterval/getClassData.cc
<TargetRTS>/src/RTMessage/internData.cc
<TargetRTS>/src/RTObject_class/abstract.cc
<TargetRTS>/src/RTObject_class/bool.cc
<TargetRTS>/src/RTObject_class/char.cc
<TargetRTS>/src/RTObject_class/double.cc
<TargetRTS>/src/RTObject_class/float.cc
<TargetRTS>/src/RTObject_class/int.cc
<TargetRTS>/src/RTObject_class/long.cc
<TargetRTS>/src/RTObject_class/nop.cc
<TargetRTS>/src/RTObject_class/pvoid.cc
<TargetRTS>/src/RTObject_class/short.cc
<TargetRTS>/src/RTObject_class/string.cc
<TargetRTS>/src/RTObject_class/uchar.cc
<TargetRTS>/src/RTObject_class/ulong.cc
<TargetRTS>/src/RTObject_class/unsigned.cc
<TargetRTS>/src/RTObject_class/unsignedchar.cc
<TargetRTS>/src/RTObject_class/unsignedint.cc
<TargetRTS>/src/RTObject_class/unsignedlong.cc
<TargetRTS>/src/RTObject_class/unsignedshort.cc
<TargetRTS>/src/RTObject_class/ushort.cc
<TargetRTS>/src/RTObject_class/void.cc
<TargetRTS>/src/RTPointer/getClassData.cc
<TargetRTS>/src/RTPriority/getClassData.cc
<TargetRTS>/src/RTProtocol/send.cc
<TargetRTS>/src/RTProtocol/sendAt.cc
<TargetRTS>/src/RTProtocol/sendOne.cc
<TargetRTS>/src/RTReal/getClassData.cc
<TargetRTS>/src/RTSample/sample1.cc
<TargetRTS>/src/RTSample/sample2.cc
<TargetRTS>/src/RTSample/sample3.cc
<TargetRTS>/src/RTSequence/getClassData.cc
<TargetRTS>/src/RTSequenceOf/getClassData.cc
<TargetRTS>/src/RTSignal/getClassData.cc
<TargetRTS>/src/RTString/ct.cc
<TargetRTS>/src/RTString/getClassData.cc
<TargetRTS>/src/RTTime/getClassData.cc
<TargetRTS>/src/RTTimerId/classData.cc
<TargetRTS>/src/RTTimespec/classData.cc
<TargetRTS>/src/RTUnknownObject/classData.cc
<TargetRTS>/src/RTWrapper/classData.cc
```

Setting timers with std::chrono library types

The API provided in `RTTiming.h` was extended to allow timers to be set using types from the std::chrono library. At the same time the GCC flag `-Wuseless-cast` was removed since the new implementation uses casts that are needed on some platforms (e.g. Windows), but not needed on others (e.g. Linux).

Changed the following files:

```
<TargetRTS>/include/RTTiming.h
<TargetRTS>/include/RTTiming.inl
<TargetRTS>/src/RTTiming/informAt.cc
<TargetRTS>/libset/x64-MinGw-8.1.0/libset.mk
<TargetRTS>/libset/x64-gcc-4.x/libset.mk
<TargetRTS>/libset/x64-gcc-7.x/libset.mk
```

Changes related to how evtState is emitted

Previously the observability event evtState was emitted at the end of executing a transition. This was changed to instead emit it when a state is entered, just before running the entry action of the state (if any). This change was made to address various issues in the user interface related to state animation and tracing.

Added the following files:

```
+<TargetRTS>/src/RTActorProbe/evtState.cc
+<TargetRTS>/src/RTActorProbe/setState.cc
+<TargetRTS>/src/RTObserver/evtState.cc
+<TargetRTS>/src/RTToolsetObserver/evtState.cc
```

Changed the following files:

```
<TargetRTS>/src/include/RTActorProbe.h
<TargetRTS>/src/include/RTObserver.h
<TargetRTS>/src/include/RTToolsetObserver.h
<TargetRTS>/src/include/endeBoth.h
<TargetRTS>/src/include/endePriv.h
<TargetRTS>/src/include/endeTarg.h
<TargetRTS>/src/MANIFEST.cpp
<TargetRTS>/src/RTActor/enterState.cc
<TargetRTS>/src/RTActorProbe/evtChain.cc
<TargetRTS>/src/RTActorProbe/ct.cc
<TargetRTS>/src/RTActorProbe/dt.cc
<TargetRTS>/src/RTActorProbe/unused.cc
<TargetRTS>/src/RTObserver/dispatchEvent.cc
<TargetRTS>/src/RTToolsetObserver/endeBoth.cc
<TargetRTS>/src/RTToolsetObserver/endeTarg.cc
```

Added support for Visual Studio 2022

The TargetRTS now contains new config and libset files and prebuilt libraries for supporting compilation with Visual Studio 2022 (ver 17.0), for both 32 (x86) and 64 bit (x64).

Added the following folders and files:

```
+ <TargetRTS>/config/WinT.x64-VisualC++-17.0/
+ <TargetRTS>/config/WinT.x86-VisualC++-17.0/
+ <TargetRTS>/libset/x64-VisualC++-17.0/
```

```
+ <TargetRTS>/libset/x86-VisualC++-17.0/
+ <TargetRTS>/lib/WinT.x64-VisualC++-17.0/
+ <TargetRTS>/lib/WinT.x86-VisualC++-17.0/
```

Removed the following folders and files:

```
- <TargetRTS>/config/WinT.x64-VisualC++-15.0/
- <TargetRTS>/config/WinT.x86-VisualC++-15.0/
- <TargetRTS>/libset/x64-VisualC++-15.0/
- <TargetRTS>/libset/x86-VisualC++-15.0/
- <TargetRTS>/lib/WinT.x64-VisualC++-15.0/
- <TargetRTS>/lib/WinT.x86-VisualC++-15.0/
```

Added support for MinGW 12.2.0

The TargetRTS now contains new config and libset files and prebuilt libraries for supporting compilation with MinGW 12.2.0 for 64 bit (x64).

Added the following folders and files:

```
+ <TargetRTS>/config/WinT.x64-MinGw-12.2.0/
+ <TargetRTS>/libset/x64-MinGw-12.2.0/
+ <TargetRTS>/lib/WinT.x64-MinGw-12.2.0/
```

Removed the following folders and files:

```
- <TargetRTS>/config/WinT.x64-MinGw-8.1.0/
- <TargetRTS>/libset/x64-MinGw-8.1.0/
- <TargetRTS>/lib/WinT.x64-MinGw-8.1.0/
```

Fix memory leak in evtMessageQueue

A memory leak was observed in the file `evtMessageQueue.cc`. This memory leak was due to improper release of memory for `msgDataBuf` using delete. The memory is allocated for char array and is now freed by using delete[].

Changed the following files:

`<TargetRTS>`/src/RTToolsetObserver/evtMessageQueue.cc

Moving data when sending event on replicated port

If you send an event on a replicated port (i.e. a port with multiplicity > 1) without specifying a port index, the event will be broadcasted, i.e. sent on all port instances. The data of the event will by default be copied once each time the event gets sent on a port instance. However, if you specify that data instead should be moved, then this behavior must change since it’s only possible to move the event data once. Previously there was an attempt to move the event data multiple times, but now it has been corrected so that the event data only will be moved once, namely for the last port instance. For all other port instances the data will be copied.

Changed the following files:

`<TargetRTS>`/src/RTProtocol/send.cc

Added support for gcc 12.x

The TargetRTS now contains new config and libset files and prebuilt libraries for supporting compilation with gcc-12.x for 64 bit (x64) and removed precompiled library for gcc-4.x and gcc-7.x

Added the following folders and files:

```
+ <TargetRTS>/config/LinuxT.x64-gcc-12.x/
+ <TargetRTS>/libset/x64-gcc-12.x/
+ <TargetRTS>/lib/LinuxT.x64-gcc-12.x/
```

Removed the following folders and files:

```
- <TargetRTS>/config/LinuxT.x64-gcc-7.x/
- <TargetRTS>/libset/x64-gcc-7.x/
- <TargetRTS>/lib/WinT.x64-gcc-7.x/
- <TargetRTS>/config/LinuxT.x64-gcc-4.x/
- <TargetRTS>/libset/x64-gcc-4.x/
- <TargetRTS>/lib/WinT.x64-gcc-4.x/
```

Changed the following files:

```
<TargetRTS>/include/RTMain.h
<TargetRTS>/src/RTMain/installHandlers.cc
<TargetRTS>/src/RTMain/installOneHandler.cc
<TargetRTS>/src/target/win/RTMain/installOneHandler.cc
```

New debugger command SendToApp

The trace UI provides a new button “Send to App” which can send an arbitrary text message to the TargetRTS. A corresponding debugger command is now implemented by the TargetRTS. By default only two internally used text messages are handled by the command (rtLogAll and rtDelAll) but you can add handling of your own custom text messages in a similar way. This can for example be useful for tracing some information to the console when debugging an application.

Added the following folders and files:

```
+ <TargetRTS>/src/RTDaemon/deleteActorInfo.cc
+ <TargetRTS>/src/RTDaemon/logActorInfo.cc
+ <TargetRTS>/src/RTToolsetObserver/handleSendToApp.cc
```

Changed the following files:

```
<TargetRTS>/src/MANIFEST.cpp
<TargetRTS>/src/include/RTDaemon.h
<TargetRTS>/src/include/RTDaemonInfo.h
<TargetRTS>/src/include/RTToolsetObserver.h
<TargetRTS>/src/include/endeBoth.h
<TargetRTS>/src/include/endePriv.h
<TargetRTS>/src/RTDaemon/eventMatches.cc
<TargetRTS>/src/RTDaemon/posInfo.cc
<TargetRTS>/src/RTToolsetObserver/encoDeco.cc
```

```
<TargetRTS>/src/RTToolsetObserver/endeBoth.cc
<TargetRTS>/src/RTToolsetObserver/endeTarg.cc
```

Handling messages received before capsule initialization

It's possible to construct a model where (under certain special circumstances) a capsule may receive a message before its state machine has been initialized. Previously this led to a call to RTActor::unexpectedMessage(), i.e. this situation was considered unexpected. Now this situation is handled by deferring such messages and later recalling them when the capsule is ready to initialize its state machine.

Added the following folders and files:

```
+ <TargetRTS>/src/RTActor/messageReceivedBeforeInitialized.cc
+ <TargetRTS>/src/RTActor/unhandledMessage.cc
```

Changed the following files:

```
<TargetRTS>/src/MANIFEST.cpp
<TargetRTS>/include/RTActor.h
<TargetRTS>/src/include/RTRecallFilter.h
<TargetRTS>/src/RTMessage/deliver.cc
<TargetRTS>/src/RTRecallFilter/ct.cc
<TargetRTS>/src/RTRecallFilter/select.cc
<TargetRTS>/src/RTRecallFilter/unused.cc
```

Remove using namespace std

According to C++ coding standards we should not use using namespace std. Instead of this we should use std:: or using declaration for all variables or function defined in standard namespace because it can leads to ambiguities if we will use other lib with TargetRTS having same name.

Changed the following files:

```
<TargetRTS>/src/RTToolsetObserver/evtMessageQueue.cc
```

Added move constructor for RTTypedValue_RTString struct

A new constructor was added with the following signature for RTTypeValue_RTString struct to support move semantics.

```
inline RTTypedValue_StdString(const StdString&& rtg_value, const RTObject_class* rtg_type)
    : data(&rtg_value)
    , type(rtg_type)
    , rValueRef(true)
```

Changed the following files:

```
<TargetRTS>/include/RTString.h
<TargetRTS>/include/RTString.inl
```

Added support for Clang 16.x on Windows

The TargetRTS now contains new config, libset files and prebuilt libraries for supporting compilation with Clang-16.x for 64 bit (x64) Windows operating system.

Added the following folders:

```
+ <TargetRTS>/config/WinT.x64-Clang-16.x/
+ <TargetRTS>/libset/x64-Clang-16.x/
+ <TargetRTS>/lib/WinT.x64-Clang-16.x/
```

Changed the following files:

```
<TargetRTS>/src/RTProtocol/reply.cc
<TargetRTS>/src/RTinet/bind.cc
<TargetRTS>/src/RTinet/connect.cc
<TargetRTS>/src/Rtinet/getname.cc
```

Added support for Clang 15.x on VxWorks

The TargetRTS now contains new config, libset files and prebuilt libraries for supporting compilation with Clang-15.x for SIMNT CPU on VxWorks7 RTOS. WindRiver provides Clang-15.x support on VxWorks7 so this support is done by installing VxWorks7 on Windows machine.

Added the following folders:

```
+ <TargetRTS>/config/VxWorks7T.simnt-Clang-15.x/
+ <TargetRTS>/libset/simnt-Clang-15.x/
+ <TargetRTS>/lib/VxWorks7T.simnt-Clang-15.x/
+ <TargetRTS>/src/target/VxWorks7T
+ <TargetRTS>/target/VxWorks7T
```

Changed the following files:

```
<TargetRTS>/src/RTProtocol/reply.cc
```

Compiling the TargetRTS with a compiler not supporting C++ 11

The TargetRTS can now be compiled with an old C++ complier that doesn’t support C++ 11. As part of this change the macro RTUseSTL has been removed since different versions of C++ include different versions of STL. Now the macro __cplusplus is used for conditional compilation required due to different C++ versions. Note that it’s still recommended to use a C++ 11 compiler, if possible, since with an older compiler not all features are available. If you happen to use an Model RealTime feature that requires a more modern compiler than what you use, you will either get an error during code generation or when compiling the generated code.

Changed the following files:

```
<TargetRTS>/include/RTActorFactory.h
<TargetRTS>/include/RTConfig.h
<TargetRTS>/include/RTDynamicStringOutBuffer.h
<TargetRTS>/include/RTFrame.h
<TargetRTS>/include/RTInjector.h
<TargetRTS>/include/RTString.h
<TargetRTS>/include/RTString.inl
<TargetRTS>/include/RTTiming.h
<TargetRTS>/include/RTTiming.inl
<TargetRTS>/src/MANIFEST.cpp
<TargetRTS>/src/RTActorFactory/ct.cc
<TargetRTS>/src/RTDaemon/logActorInfo.cc
<TargetRTS>/src/RTString/ct.cc
<TargetRTS>/src/RTString/getClassData.cc
<TargetRTS>/src/RTTiming/informAt.cc
<TargetRTS>/src/include/RTDaemonInfo.h
```

Added the following file:

```
+ <TargetRTS>/include/RTTimingChrono.inl
```

Add support for long long and unsigned long long primitive

The TargetRTS now contains support for long long and unsigned long long primitive data types.

Added the following files:

```
+ <TargetRTS>/src/RTAsciiDecoding/get_unsignedlonglong.cc
+ <TargetRTS>/src/RTAsciiEncoding/put_long_long.cc
+ <TargetRTS>/src/RTAsciiEncoding/put_unsignedlonglong.cc
+ <TargetRTS>/src/RTFormat/longlong.cc
+ <TargetRTS>/src/RTFormat/unsignedlonglong.cc
+ <TargetRTS>/src/RTLog/log_longlong.cc
+ <TargetRTS>/src/RTLog/log_unsignedlonglong.cc
+ <TargetRTS>/src/RTLog/show_longlong.cc
+ <TargetRTS>/src/RTLog/show_unsignedlonglong.cc
+ <TargetRTS>/src/RTObject_class/longlong.cc
+ <TargetRTS>/src/RTObject_class/unsignedlonglong.cc
+ <TargetRTS>/src/RTTypedValue/longlong.cc
+ <TargetRTS>/src/RTTypedValue/unsignedlonglong.cc
```

Changed the following files:

```
<TargetRTS>/include/RTAsciiDecoding.h
<TargetRTS>/include/RTAsciiEncoding.h
<TargetRTS>/include/RTDecoding.h
<TargetRTS>/include/RTEncoding.h
<TargetRTS>/include/RTFormat.h
<TargetRTS>/include/RTLog.h
<TargetRTS>/include/RTObject_class.h
```

```
<TargetRTS>/src/MANIFEST.cpp
<TargetRTS>/src/RTAsciiDecoding/get_long_long.cc
```

Added support for Clang 14.x on MacOS

The TargetRTS now contains new config, libset files and prebuilt libraries for supporting compilation with Clang-14.x for 64 bit (x64) Mac operating system.

Added the following folders:

```
+ <TargetRTS>/config/MacT.x64-Clang-14.x/
+ <TargetRTS>/libset/x64-Clang-14.x/
+ <TargetRTS>/lib/MacT.x64-Clang-14.x/
+ <TargetRTS>/src/target/mac/
```

Changed the following files:

```
<TargetRTS>/src/Build.pl
<TargetRTS>/src/Makefile
```

Improve Dependency Injection for Optional Parts

The dependency injection service (see Capsule factories and dependency injection) was improved so that if no create function is registered for a capsule part that gets incarnated, the capsule instance will now be created in the default way. Previously an instance of the capsule that types the capsule part was created by the RTInjector, but for optional capsule parts it's better to instead create the instance using the arguments provided in the call to `incarnate()`. This improvement avoids the need to register create functions for all optional capsule parts where the type of the created capsule instance is different from the type of the capsule part.

Changed the following files:

```
<TargetRTS>/include/RTInjector.h
<TargetRTS>/src/RTActorRef/incarnate2.cc
<TargetRTS>/src/RTInjector/create.cc
```

Version 8.0.0 and Model RealTime 12.0

Version 12.0 of Model RealTime includes Code RealTime which is a new product for developing realtime C++ applications in Visual Studio Code and Eclipse Theia. The TargetRTS is now used by both these two products, and it was therefore decided to uplift its version to 8.0.0 to clearly make a difference to the older versions. Note that exactly the same TargetRTS is included both with Model RealTime and Code RealTime which makes it possible to use them together when building a realtime application.

Changed the following files:

```
<TargetRTS>/include/RTVersion.h
```

NOTE: Future changes to the TargetRTS will be documented here: https://secure-dev-ops.github.io/code-realtime/target-rts/changelog/