

QNX Momentics használati útmutató a MS Visual Studio felhasználói számára

Szerző: Dennis Kelly

Az útmutató célja

Mivel a QNX Momentics fejlesztőkészlet ingyenesen letölthető és tesztelhető, az alábbi dokumentáció megkísérli a Windows Visual Studio felhasználói számára bemutatni a QNX Momentics IDE fejlesztőkörnyezetet. Az útmutató segítségével a Windows programozók könnyebben sajátíthatják el a QNX Momentics használatát.

Terminológia

A dokumentáció a “Visual Studio”, a “Studio” és a “VC++” kifejezéseket felváltva használja.

A dokumentáció a “Visual Studio 6” és a “Visual Studio 2003/2005”, verziókon alapul, a verziók között nem tesz különbséget. Az útmutató feltételezi a cikk olvasójának jártasságát valamely említett verziójú fejlesztőrendszerrel kapcsolatban.

Bevezetés

A QNX Momentics IDE többféle operációs rendszerhez is támogatott. Létezik QNX, Linux, Solaris és Windows alatt futtatható változat is.

A QNX Momentics alatti programfejlesztés és hibakeresés (debug) módszere néhány alapvető szempontban is eltér a szokásos Windows környezetben megszokottól. A legfontosabb eltérés akkor jelentkezik, ha nem QNX, hanem más operációs rendszer alatt, pl. Windows gazdagépen futtatjuk a Momentics-et:

A felhasználó által megírt program csak egy másik számítógépen futtatható és/vagy tesztelhető, amely gép nem azonos a számítógéppel, amelyen a fejlesztés történik. A fejlesztői gép neve „host”, a futtató gép elnevezése “target”

- Amikor egy Win32 programot lefordítunk a VC++-al, az a program tipikusan ugyanazon a PC-n tesztelhető és futtatható, amelyen azt elkészítettük.
- Amikor a QNX Momentics-et Windows alatt használjuk, az elkészült kód nem futtatható, nem tesztelhető sem ugyanazon a gépen, sem Windows alatt.

A QNX Momentics egy “cross-compiler”, amellyel másfajta processzor architektúrára is lefordíthatunk programokat, amelyeket szintén a QNX6 (Neutrino) operációs rendszert futtat.

- A target gép **lehet** x86, de lehet PowerPC vagy ARM processzoros is.
- **Ha** a mi target-ünk x86, abban az esetben egy **virtuális PC-t** is alkalmazhatunk target-ként, mint pl. a VMware.

- **Ha** a mi target-ünk NEM x86, abban az esetben is használhatunk **virtuális PC-t** a complex algoritmusok tesztelésére, mielőtt a target-en véglegesítenénk a szoftvert.

Ez a megoldás portábilis kódot, tehát több processzor platformon is alkalmazható szoftvert eredményez. Viszont ez a cross-compile megoldás egy másfajta, új szemléletet követel meg. A fejlesztőrendszer (IDE) ismerete is komplexebb ismereteket kíván. Ez az új, eltérő szemlélet bemutatása a jelen dokumentáció alapvető célja.

A Target számítógép előkészítése

A QNX Momentics Windows host-on történő futtatása azt jelenti, hogy a fejlesztői környezettől **minden esetben** eltérő, más géphez fogunk programot fejleszteni és tesztelni.

A Momentics-et egészen addig nem tudjuk használni, amíg nem építettünk ki kapcsolatot egy megfelelő target-hez. Ez a legelső feladat, amelyet teljesítenünk kell.

A legegyszerűbb megoldás a VMware (vagy hasonló termék) használata, a QNX Neutrino operációs rendszer futtatásához.

- 1) Hozzunk létre egy új, üres virtuális gépet a VMware alatt. Nem kell nagy lemezterületet kijelölnünk, kb. 500MB méretű virtuális partíció elég lesz.
- 2) Irjuk ki a Neutrino runtime ISO file-t CD-re, majd installáljuk a QNX operációs rendszert erről a CD-ről, a virtuális gépre.
- 3) Engedélyezzük a “private shared network”-öt a Windows alatt.
- 4) Állítsuk be a processzorok számát “2”-re, ha dual core típusú PC-nk van.
- 5) Az installáció elvégzése után a login név legyen “root” (password nélkül), a target gépen.
- 6) A jobb egérgombbal kattintva válasszuk ki a “Terminal”-t.
- 7) A terminal ablakban el kell indítanunk a **qconn** elnevezésű programot. Ez a programocska lesz a "QNX Momentics target debug agent", amely képes a windows oldali Momentics-el kapcsolatot tartani. Gépeljük be a parancsot a terminál ablakba, majd üssünk enter-t.
- 8) Gépeljük be: “**ifconfig**”, ahhoz, hogy láthassuk az IP címet. Másik lehetőség, ha a képernyőablakban jobb oldalán klikkelünk a **Configure --> Network** lehetőségre a gombok közül.
A virtuális gép IP címét mindenképp ismernünk kell. Nélküle nem tudunk csatlakozni a windows felőli Momentics-el, a target-hez.
- 9) Ping-eljük meg Windowsból a QNX6 virtuális gépet, majd a QNX6 felől a Windows-t is. Ezzel ellenőrizhetjük az IP csatlakozást.

Ezzel már fut a target, valamint a debug agent is (“qconn”), valamint ismerjük a target IP címét, képesek leszünk futtani és debug-olni a legelső megírt programunkat.

Program készítése és tesztelése a QNX Momentics-el

A QNX Momentics alatti programkészítés nagyon hasonló a VC++ alatti munkához. Mindkét környezetben varázslót is használhatunk az új project típusának meghatározásához, vázának létrehozásához.

Ahogy a Visual Studio-ból ez már ismert, a Momentics-nek is a munkaterület, a ‘**workspace**’ a koncepciója. A workspace több független ‘**project**’-et is tartalmazhat.

Egy különbség mégis van a Momentics és a VC++ között.

A workspace és a project-ek könyvtár alapúak. Más szóval, a workspace a könyvtár gyökere, amely több alkönyvtárat (folders) is tartalmazhat, a projekt-ek neveivel azonosítva. A meta-adatok xml file-ban vannak tárolva, **.project** névvel, minden projekt könyvtárban.

(A VC++ workspace-ekben, a projekteknek nem kell a saját könyvtáraikban elhelyezkedniük, csak jelen kell lenniük a bal oldali navigációs panel fa struktúrájában.)

Amikor elindítjuk a Momentics-et, *meg kell* adnunk a workspace nevét, amelyet használni akarunk. (enélkül egy alapértelmezettet nyit meg). Ha nincs ilyen, akkor egy üres workspace lesz létrehozva.

Ha a Momentics workspace meg van nyitva, akkor az IDE felület nagyon ismerős lesz a Visual Studio felhasználói számára.

- A bal oldali függőleges panel egy fa struktúrát tartalmaz, amely a workspace projektjeit mutatja meg.
- A fa struktúra jobb oldalán a szerkeszthető forrásfájlok helyezkednek el.
- Az alsó panel az üzenetek számára van nyitva

Most már elkészíthetjük első projektünket a megnyitott workspace-ben. Válasszuk a **File --> New --> Project** –et a felső menüből. Az elindult varázslóban a ‘C’ folder alatt, a “QNX C Project”-et kell választani. Ezután nyomjuk meg a ‘Next’-et, majd írjuk be a projekt nevét. Ez például, “try1” is lehet. Mi a továbbiakban ezt a nevet fogjuk használni.

A ‘Next’ gombbal léphetünk tovább.

Mivel a Momentics keresztfordítást (cross-compile) fog alkalmazni a projektünk lefordítására, még a projekt elkészítése előtt **ki kell** választanunk a “Build Variants” fület a párbeszéd ablakon. *Itt fogjuk kiválasztani a target processzor típusát, valamint meg kell határoznunk, hogy a lefordított kód debug és/vagy kiadásra(telepítésre) szánt verzió lesz-e.* Esetleg az összes processzortípusra is *lehetséges* a programunk lefordítása, bár valószínűleg ez csak feleslegesen hosszabítani fogja a program lefordításának idejét.

Ezután után klikkeljünk a “Finish” gombra, ezzel elkészült a QNX C projekt, benne a szokásos “Hello, World” program forráskódja.

Az IDE képességeinek bemutatásához, kettő vagy több projekt kell. Ezért ismételjük meg a fenti lépéseket, készítsünk egy második projektet. Most válasszuk a “QNX C++ Project” lehetőséget. A második projekt neve legyen “try2”, a példa miatt.

Megjegyzés a C és C++ -ról:

QNX6 alatt a legtöbb program valószínűleg C nyelven lesz megírva, nem C++ -ban. Nagy valószínűséggel a QNX6-ban megírt szoftverek többsége a „rendszer”-hez lesz közelebb, ezért a C++ valamelyest kevésbé támogatott. De ez **NEM** a compiler szintjén jelentkezik, hanem inkább a kényelmi szolgáltatásokban.

Amennyiben C++ -ban írunk programot QNX6 alá, **a gcc compiler teljesen támogatott** a Momentics-ben. Bár itt meg kell jegyezni, hogy amikor az operációs rendszer felé adunk ki egy hívást, abban az esetben C függvényhívást kell realizálnunk. *Ez a Windows esetében is teljesen így működik. Amikor a Win32 SDK-t használjuk, a helyzet pontosan ugyanaz.*

A rendszer szoftverek hagyományosan C nyelven íródnak. Ez a QNX6-ra is igaz, ahogyan a Linux-ra is, valamint bármilyen *NIX-típusú operációs rendszer esetében is ez a helyzet.

Megj: A C++ program moduloknak Momentics-ben a kiterjesztése “.cc”, alapértelmezésben.

Ahogyan a VC++ esetében is, a Momentics IDE –ben le kell fordítani a projektet, mielőtt a hibakeresést (debug), az egylépéses végrehajtást, stb elkezdhetnénk.

A legegyszerűbb megoldás az, ha a bal oldali panelen a projekt nevére kattintunk a jobb egérgombbal, aztán kiválasztjuk a “Build Project” lehetőséget. Ezt a módszert alkalmazva a projekteket külön-külön, egyesével is lefordíthatjuk.


Ez a megoldás teljesen azonos azzal, ha először kiválasztjuk a bal panelen a fordítani kívánt projektet, majd kiválasztjuk a **Project Build Project** –et a felső menüből.

A második módszer pontosan azt mutatja be, hogy az IDE rendszer érzékeny arra, melyik projekt van kiválasztva a fa struktúrát megjelenítő panelen. Más szóval, az aktív projektet ki kell választani a Momentics fa struktúráján belül. Például, a C/C++ nézet váltás után a Console megjelenik alul, ezzel lehetővé válik a kiválasztott projekt fordításának eredménye.

A Visual Studio 6 esetében ki kell választani az “Active Project”-et a legördülő menüből, amely egy listát tartalmaz az összes projekt neveiről, melyek a vannak jelen a workspace-n belül. Ez a választás fogja kijelölni a választott projektet a fa nézetben. Amikor kiválasztjuk a Build (F7) opciót, lefordul és összeszerkesztésre (build) kerül a kijelölt, csak a kiválasztott projekt.

(a későbbi verziókban hasonlóan, a ‘Configuration Manager’-ben kell kiválasztani a projektet.)



A build elvégzéséhez a másik megoldás az eszköztár  jelű gombjának használata. A gomb megnyomása után az összes projektből elkészül a futtatható verzió, nem csak a kijelölt projektből.

Jegyezzük meg, hogy a 'Console output panel' *csak az aktív projekt szerkesztés (build) tájékoztató üzeneteit tartalmazza. Ahhoz, hogy a többi projekt build-jének az üzeneteit is megtekinthessük, a megfelelő projektet ki kell választani a fa nézetben.*

Bármilyen build művelettel kapcsolatosan a "Problems" fülre kattintva jelennek meg a fordítás figyelmeztető üzenetei és a hibák (compile warnings and errors) az **összes projektről** – nem csak a kiválasztott projektről!

Program hibakeresés a cél számítógépen (debugging on target)

A "Hello, world" program ezzel elkészült, a QNX Neutrino operációs rendszer fel van installálva és kommunikál a Windows-al. Most elkezdhetjük a hibakeresési munkát.(debug session)


A Visual Studio esetében, amikor a hibakeresés a feladat, a munkaterület, (workspace) átvált egy másik konfigurációra. A Studio-ban amikor kiválasztjuk a "Start Debugging"-ot, az eszköztár megváltozik a panelek elrendezésével együtt. Amikor kiválasztjuk a "Stop Debugging"-ot, az elrendezés visszavált az eredetire.

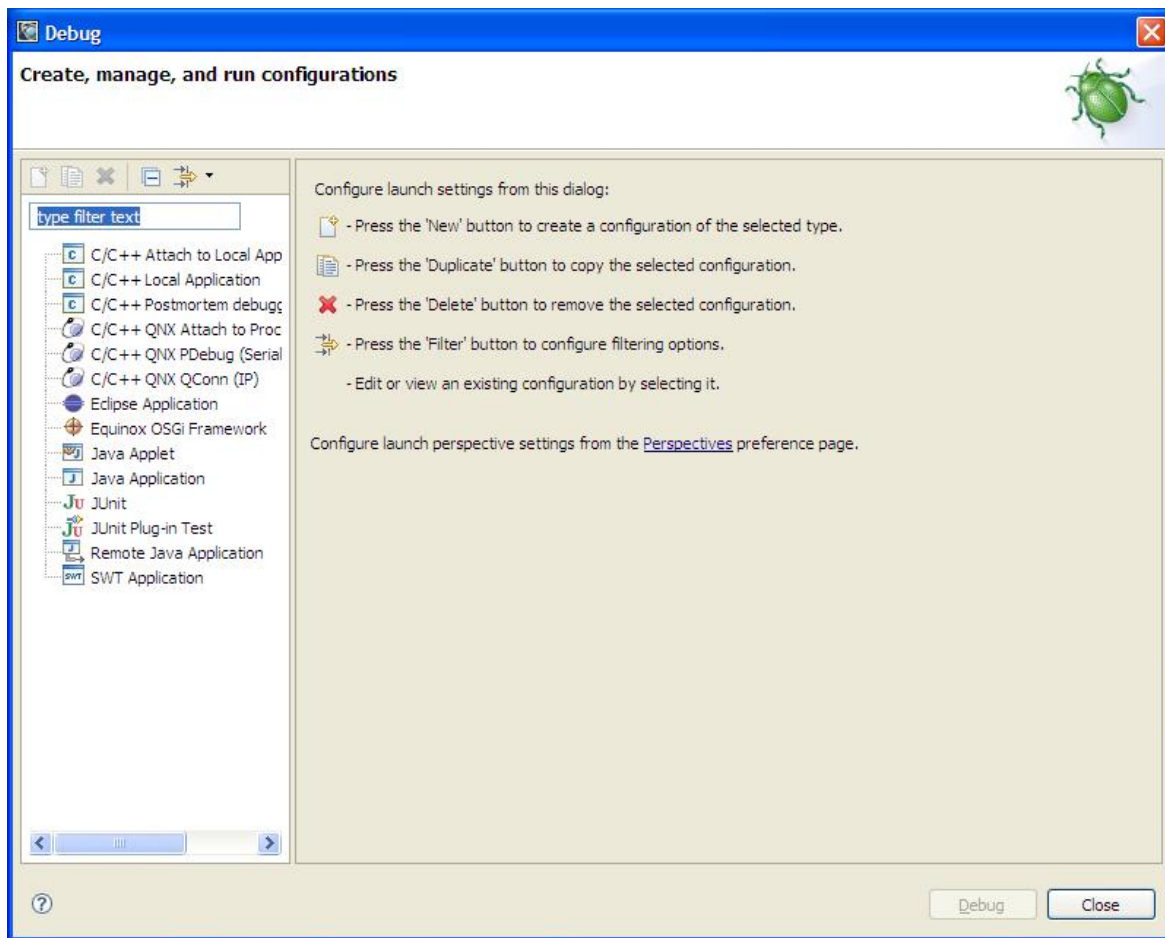
A Momentics esetében, egy hasonló dolog történik, amikor elkezdjük a debuggolást. A Momentics terminológiában, az IDE **nézetet vált**, a "C/C++"-ról (ez az alapértelmezett nézet megnyitáskor) a "Debug" nézetre.

*Az eltérés az, hogy míg a Visual Studio-ban kétféle nézet szokásos, addig a Momentics-ben sokféle van. A nézetek listáját megtekinthetjük a felső menüből kiválasztott **Window-->Open Perspective-->Other** úton. Mivel a Momentics nagyon jól konfigurálható, a teljes nézetlista attól is függ, milyen további kiegészítések, plugin-ok vannak felinstallálva a Momentics IDE alá.*


A Visual Studio-ban, a "debug" nézethez el kell kezdeni a program debuggolását is. Momentics-ben, kiválaszthatunk bármilyen nézetet, még a Debug nézetet is, bármikor. Tipikusan, a jobb oldalon található legfelső fülkészlet minimálisan a "C/C++" és a "Debug" nézeteket tartalmazza.

A debug menet elkezdéséhez az alábbi lépéseket hajtsuk végre. Előzetesen már le kellett fordítanunk a projektjeinket:

1. A legfelső menüből válasszuk a **Run -->Debug...** vagy kattintsunk a  ikonra az eszköztáron, az alábbi Debug ablak megnyitásához:



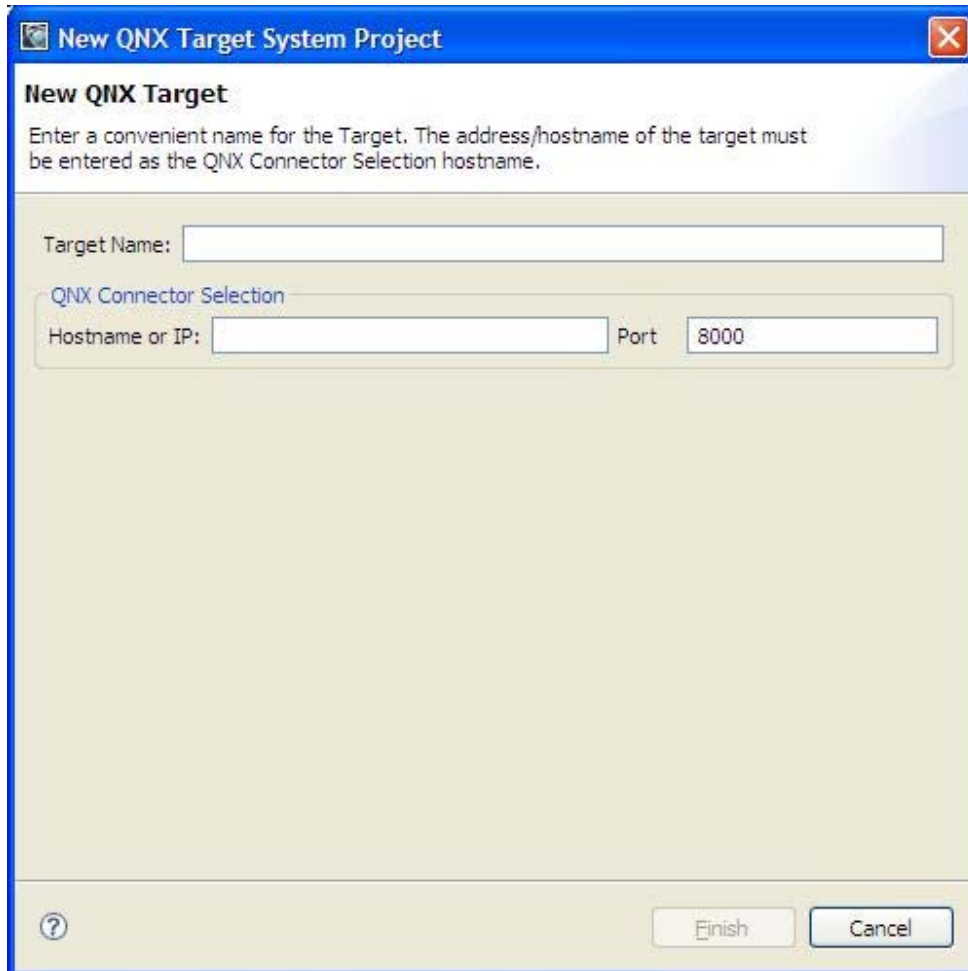
A projekt konfiguráció beállítása mindenképp szükséges, ahhoz hogy a Debug működhessen. A futtatható bináris állomány és a target gép jellemzőit itt állíthatjuk be. Ezt “Run Configuration”-nek nevezzük.

2. Az indításhoz, válasszuk a “C/C++ QNX QConn (IP)” –t a bal oldali panelen.
3. Kattintsunk a  “New” ikonra, amely a panel jobb oldalán, a legelső sorban található.

Emlékeznünk kell arra, hogy a “qconn” programnak futnia kell a target operációs rendszeren. Ez a ‘debug agent’ elnevezésű segédprogram fog kommunikálni az IP kapcsolaton keresztül a Windows és a virtuális gép között.

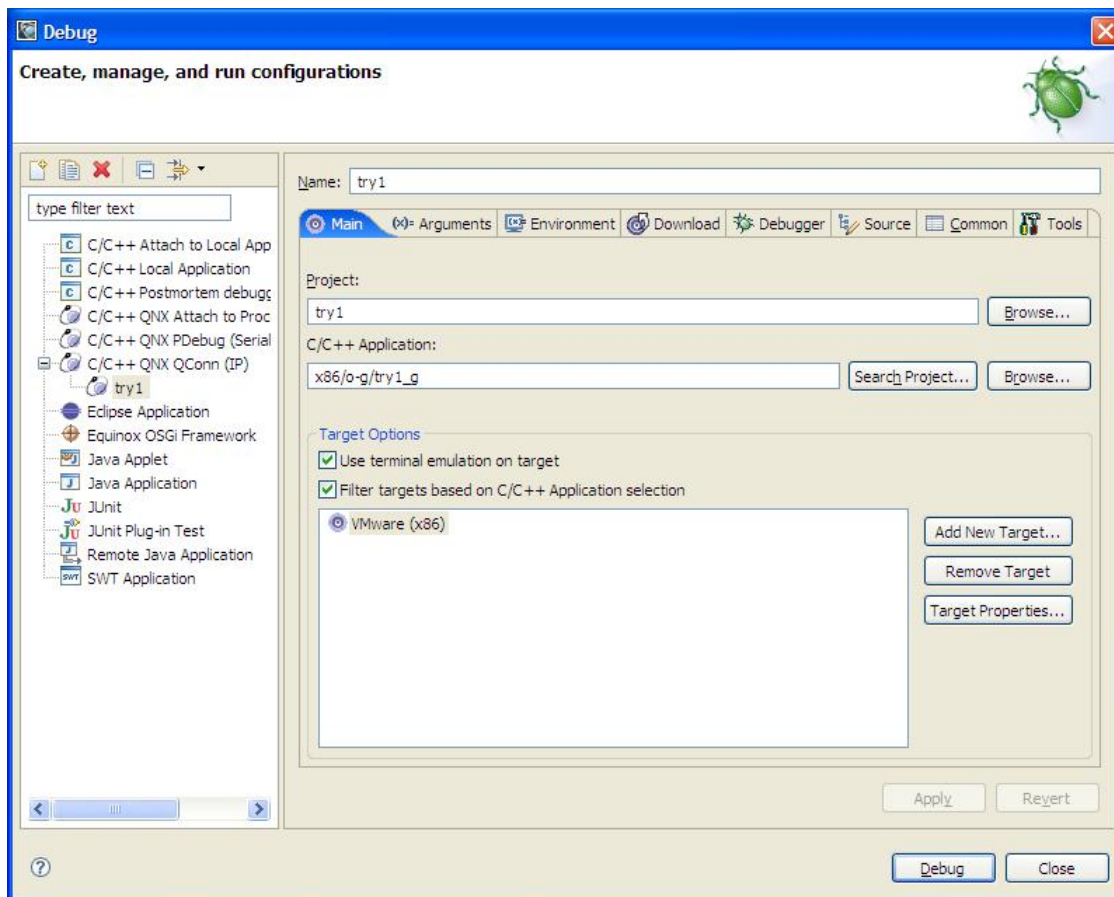
4. A “Name:” mezőbe gépeljük be: “try1”.
5. A “Project:” mező kitöltéséhez kattintsunk a Browse gombra és válasszuk: “try1”. Esetleg, a ‘try1’ projektet alapértelmezésben az alábbi útvonalon érhetjük el (az IDE v.4.01 esetében):
(C:\QNX632\ide4-workspace\try1\o-g\try1_g)
6. A “C/C++ Application:” mező kitöltéséhez a “Search Project...” gombra kattintsunk. Majd az “OK”-ra, hogy az alapértelmezettet használjuk.

7. Most pedig kattintsunk a “Add new target...” gombra, hogy a köv. dialógusablakot elérjük:

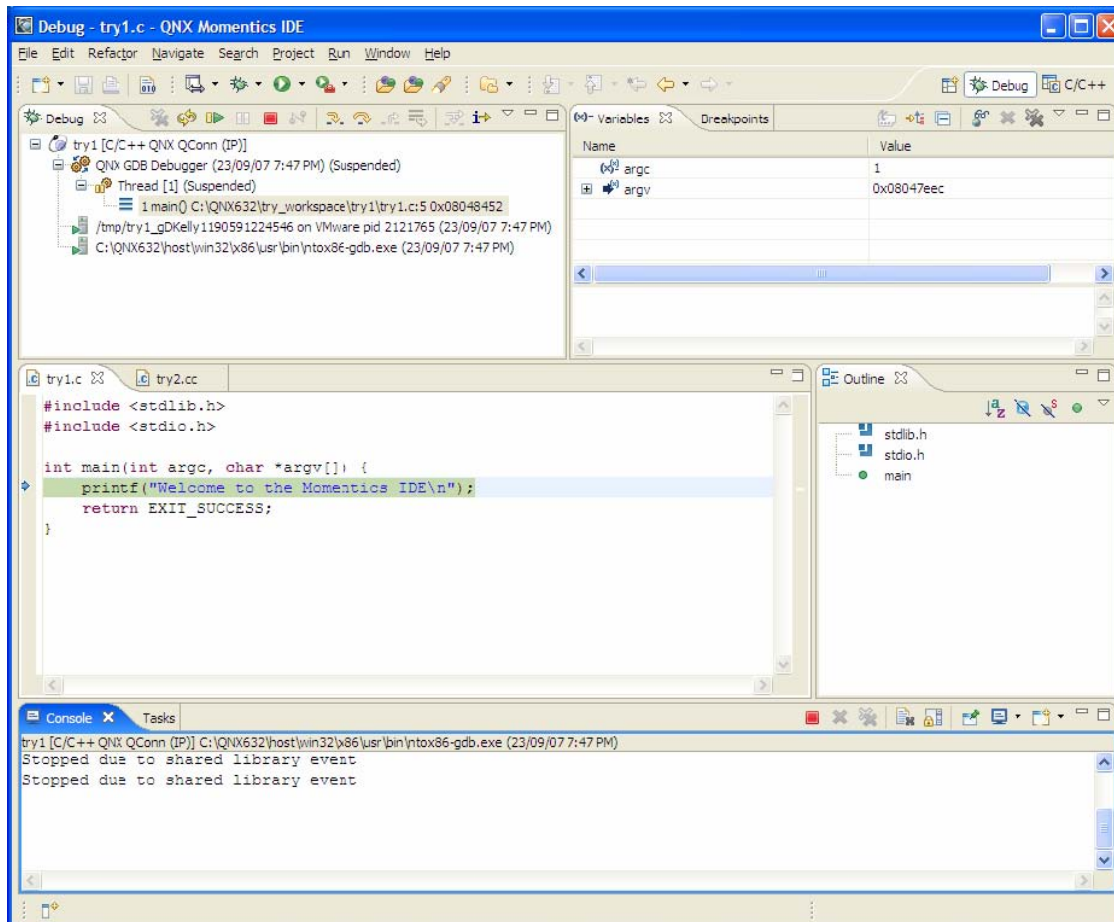




8. A “Target Name:”-hez gépeljük valamilyen nevet, pl. “VMware”.
9. A “Hostname or IP:”-hez pedig a virtuális gép címét. Az alapértelmezett TCP port szám: 8000.
10. Kattintsunk a “Finish”-re.
11. Válasszuk azt a target nevet, amelyet az előbb megadtunk (“VMware”), majd kattintsunk az “Apply”-ra.

A “Debug” nevű gomb alul most már aktívvá fog válni. A képernyőn a következő képhez hasonló helyzetet kell látnunk:



Amikor a “Debug”-ra kattintunk, a Momentics kijelzi a Debug nézetet, a megírt programunk első végrehajtható sora fénykiemelést fog kapni. A következő kép ezt az állapotot fogja mutatni.



12. A következő sorra lépéshez kattintanunk kell a  gombra, az eszköztáron. (a VC++ -ben ugyanez az F10 gombbal érhető el.) Ha a következő sor egy függvényhívás, akkor a függvény *belsejébe* belépéshez a  gombra kell kattintani. (a VC++ -ben ugyanez az F11 gombbal érhető el.)

13. A Debug funkcióbilleentyűk konfigurációjának ellenőrzéséhez vagy módosításához a felső menüből válasszuk a **Window-->Preference** -t. Azután nyissuk ki az *General/Keys*-t. Láthatjuk az alapértelmezett beállításokat, az F6 a “step over” és az F5 pedig a “step into”.


A töréspont (breakpoint) beállításához, egyszerűen egy dupla kattintással jelöljük ki a bal oldalon, a forráskódon belül a megfelelő sort ott, ahol a futtatást meg kívánjuk állítani.

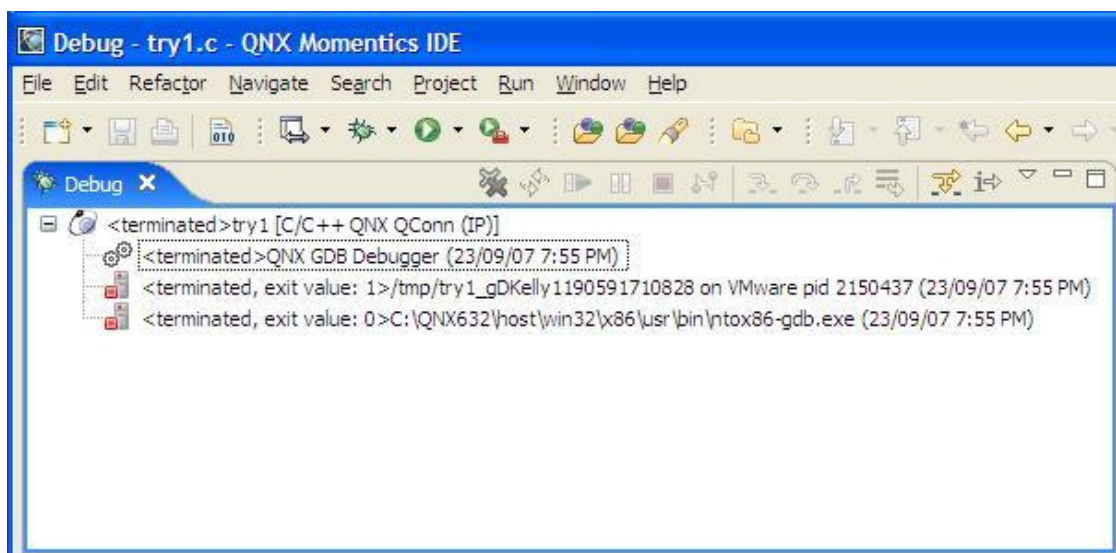
A debug folyamat menetéről a bal panel felső részén kaphatunk részletes információkat. Jellemzően nincs szükség az összes részletes információra. Azért elképzelhető, hogy hasznos lehet pontosan tudni, melyik komponens épp milyen állapotban van, hol tartunk épp az adott pillanatban:


- The executable produced by the cross-compiler was sent to the target and stored as “/tmp/try1_gDKelly1190591224546.
- The executable was started on the target with pid (process ID) 565285.
- A debug agent, “C:\QNX632\host\win32\x86\usr\bin\ntox86.exe” was started on Windows for this session to handle the communication with the target.

Ha lefuttatjuk a “pidin –p 565286” parancsot a target-en, helyettesítve a példa számot a saját debug session-ünkből vett **pid** (processz ID) azonosítóval, akkor láthatjuk is a futó processzt.

Ha a Task Manager fut a Windows alatt, akkor láthatjuk az “ntox86-gdb .exe”-t.

Ha kattintunk a  ikonra, akkor a program csak a következő töréspont-ig (breakpoint) fog futni, habár abban az esetben ha nincs egyetlen töréspont sem megjelölve, akkor természetesen folyamatosan lefut. Amikor a program futása befejeződik, a bal felső panel a következő kép szerinti állapotot fogja mutatni:



Ennél a pontnál, a target-en futó program futása normális módon fejeződött be, (második sor) a debugger komponens pedig a Windows oldalon (harmadik sor). Ha a  gombra kattintunk, akkor az ablak tartalma kitörlődik.

A Momentics valójában egy Multi-Session Debugger

A QNX Momentics IDE környezetnek van még egy további képessége, amellyel sokkal használhatóbb eszközzé vált, mint a Visual Studio.

Emlékezzünk vissza, amikor a Visual Studio debug session-je elindul, a nézet a Debug nézetre vált. A fa nézet többé nem lesz látható. Ha beleszerkesztünk a forráskódba a Debug ablakban, majd kérünk egy újrafordítást, akkor kapunk egy üzenetet: “This command will stop the

debugger.” (ez a parancs leállítja a hibakeresést). Kivétel történik, a fordítás okozta bináris kód változásai belemásolódnak a végrehajtható program állományba.

A Momentics-ben, amíg a debug session fut, visszatérhetünk a C/C++ nézetbe anélkül, hogy megzavarnánk az épp futó debug session-t. Kiválaszthatunk egy másik projektet is, ha közben azt is kell fordítani. Elkezdhetjük egy másik projekt debuggolását is – *akkor sem zavarjuk meg az eredeti debug session-t.*

Ez a multisession képesség nagyon előnyös akkor, ha két olyan processzt kell debuggolnunk, amelyek egymással kommunikálnak, pl. kliens és szerver. A Momentics-ben mindkét programfolyamat debuggolását elvégezhetjük ugyanabban az IDE környezetben. A Visual Studio esetében el kell indítani egy Studio IDE példányt a kliens, egyet a szerver program hibakereséséhez.

Amennyiben ezt a multisession képességet akarjuk kihasználni a “try1” és “try2” projektekhez, akkor egy második futtatási konfigurációt is el kell készítenünk. Előzetesen csak a “try1”-hez készítettük el. Most a “try2”-höz is el kell készítenünk a futtatási konfigurációt is.

A C/C++ nézetből, a jobb oldali egérgombbal kattintsunk a “**try2**” projektre a fa nézetben.

Válasszuk a ‘**Debug As**’-t, majd a ‘**Debug...**’-ot.

A ‘**Run Configuration**’ panel listája azt a konfigurációt fogja mutatni, amelyet már láttunk a ‘try1’ esetében. Hozzuk létre ‘**try2**’ névvel az új konfigurációt. Az előzőleg megismert módon, válasszuk a ‘**Browse**’-t a Project gomb mellett, majd válasszuk a ‘**try2**’-t.



Most a ‘**Search Project...**’ következik a ‘**C/C++ Application**’ mellett, majd válasszuk az alapértelmezettet. Fénykiemelést kell hogy kapjon a virtual PC target, majd klikkeljünk az ‘**Apply**’ és ‘**Debug**’-ra. Ezzel elkészült a ‘**try2**’ futtató konfigurációja. (ahogyan az előzőleg a ‘try1’ konfigurációja is elkészült.).



Mivel a Debug gombra kattintottunk, a nézet megváltozik a ‘Run Configuration’ panelen, a debug üzemmódoz. A try2 projekt debuggolása elindul a target-en.

Az eszköztárból, a debug ikon  melletti lefelé mutató nyíllal nyissuk le a menüjét.


Válasszuk a “**Debug...**”-ot. A bal oldali panelen láthatjuk a két projekt futtatási konfigurációját (Run Configurations) “try1” és “try2”. Mivel a “try2” már fut, ezért jelöljük ki a “try1”-et és klikkeljünk a Debug.-ra. A bal felső ablak most mutatni fogja mindkét, a debuggolás alatt lévő projektek példányait, a try1 és a try2-t.

*Az IDE környezet **egyetlen** debug vezérlő készlettel rendelkezik mindkét session számára. Fontos tudni azt, hogy **a vezérlés az épp fénykiemelt thread-hez tartozik.***

Válasszuk a Thread [1]-et a try2 fán. Nyomjuk meg a  jelű ikont, most láthatjuk hogy a try2 most “terminated”, befejezett állapotba kerül. Nyomjuk meg a  jelű ikont, a try2 példány el fog tűnni.

Most válasszuk a Thread [1]-et a try1-hez. Nyomjuk meg a  jelű ikont, a try1 példány “terminated”, befejezett állapotba kerül. Nyomjuk meg a  jelű ikont, a try1 példány el fog tűnni.

Természetesen nincs korlátozás arra, hogy egy projekthez csak egyetlen példány tartozhatna.

Ha megnyomjuk a  jelű ikont, (nem a mellette lévő, lefelé mutató nyilat) az IDE emlékezni fog a legutoljára elindított futtató konfigurációra (last Run Configuration), melyet előzőleg elindítottunk. Ebben az esetben a try1 példány jön létre. Ha még egyszer megnyomjuk, akkor ez az indítás is érvényes lesz, még egy try1 példány jön létre. Mindegyik példány debuggolható, függetlenül a vezérlőtől, mely példány kijelölt, fénykiemelt a fa nézetben.


Ez a “multisession” koncepció nagyon eltér a Visual Studio-ban megszokottól. Nagyon kényelmes munkakörnyezet, de van egy fontos dolog, amelyre nagyon figyelni kell. Amikor egy adott projektet debuggolunk, mindenképp LE KELL ÁLLITANI az előzőleg elindított példányokat. Ezt esetleg a target oldalon is megtehetjük a 'slay' parancs segítségével. Ha átváltunk a C/C++ nézetre és átszerkesztjük a forrás modult, majd lefordítjuk és újralinkeljük (rebuild), ez mindenképp LINKER ERROR-t fog okozni:

```
“cc: C:/QNX632/host/win32/x86/usr/bin/ntox86-ld caught signal 1  
cc: warning - couldn't unlink C:/QNX632/try_workspace/try1/x86/o-g/try1_g (Permission denied)”
```

Ezt a hibát csak úgy lehet megszüntetni, ha leállítjuk az **összes** előzőleg elindított, futó példányt, még azelőtt, hogy a build-et elindítanánk. Ez a Visual Studio-ban nem jöhet létre, mivel a VC++ vagy Debug módban, **vagy** build módban van. A kettő mindig kizárja egymást.

Megj: Ha kapunk egy “couldn't unlink” hibát, abban az esetben le kell állítani az összes futó példányt, még a rebuild fázis előtt.

További Momentics témakörök

Válasszuk a **Window/Open Perspective/Other-t** (vagy a -jelű ikont nyomjuk meg a Perspectives file fülön), ez felfedi a Momentics 'advanced' képességeit, amelyek implementáltak a nézetek-ben (Perspectives). Ezek a következők:

- QNX Application Profiler
- QNX Code Coverage
- QNX Memory Analysis

Ez a leírás nem tartalmazza a fenti eszközök használatát, de jól jellemzik a QNX Momentics IDE további képességeit, amelyek kiegészítő modulként (plugin-ok) lettek megírva. Érdeemes ezeket a sokoldalú eszközöket is kipróbálni.

Zárszó

Ez a dokumentum megkísérelte bemutatni, hogyan debuggolhatunk egy C/C++ programot, QNX Momentics alatt. Erre a legjobb módszer egy összehasonlítás a Visual Studio-val, így a meglévő tapasztalatokon át könnyebbé válhat a Momentics megismerése. Remélhetőleg, ez a dokumentum képes segíteni a QNX Momentics megismerésében. Sok szerencsét!

Fordította: Kovács József – RTC Automatika Kft.