Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar

Automatizálási és Alkalmazott Informatikai Tanszék
Beágyazott és irányító rendszerek szakirány – Számítógép-alapú rendszerek ágazat (BSc)
Számítógép-alapú rendszerek szakirány (MSc)

Eclipse alapú fejlesztőkörnyezet összeállítása STM32F4Discovery fejlesztőkártyához

Segédlet

Kidolgozta: Szabó Zoltán (szabo.zoltan@aut.bme.hu),
Kovács Viktor (kovacs.viktor@aut.bme.hu)

Utolsó módosítás ideje: 2012.10.05. 15:21

1. Bevezetés

Az STM32F4 sorozatú mikrokontrollerekre történő fejlesztéshez számos fejlesztőeszköz elérhető. A fejlesztőeszközök nagy részére igaz, hogy nem ingyenesek, az esetleg ingyenesen elérhető verziók pedig valamilyen korlátozást (általában generált kódméret korlátozást) tartalmaznak. Jelen dokumentum célja, hogy bemutassa a széles körben elterjedt, szabad forrású, ingyenes Eclipse alapú fejlesztőkörnyezet telepítését és használatát az STM32F4Discovery kártyához.

2. Szükséges eszközök

Eclipse IDE for C/C++ developers

Az Eclipse Java alapú, szabad forrású, széles körben használt integrált, moduláris felépítésű fejlesztőkörnyezet. Az Eclipse keretrendszerbe különböző nyelveken (JAVA, C/C++, stb.) történő szoftverfejlesztéshez kialakított modulokat tudunk telepíteni. Beágyazott rendszerekre általában C/C++ nyelven szoktunk fejleszteni, ezért nekünk az Eclipse C/C++ fejlesztéshez kialakított moduljaira van szükségünk (Eclipse IDE for C/C++ developers).

Az Eclipse IDE letölthető az alábbi linkről:
http://eclipse.org/downloads/

Törekedjünk arra, hogy mindig az elérhető legfrissebb verziót használjuk, hiszen az újabb verziók mindig egyre több hibajavítást és extra szolgáltatást tartalmaznak, ezzel nagyban megkönnyítve a fejlesztést.
Az eszköz nem rendelkezik a Windows rendszereken megszokott telepítővel, a letöltött csomagot ki kell tömöríteni és már használható is.

Sourcery CodeBench Lite Edition for ARM EABI

Az Eclipse önmagában nem elegendő a fejlesztéshez, hiszen az Eclipse csak egy fejlesztőkörnyezet. Ahhoz, hogy az Eclipse-ben elkészített forrásfáljainkat az eszközön futtatni tudjuk, szükségünk van egy fordító eszközre is (compiler toolchain). Az interneten számos gcc alapú fordító érhető el (Sourcery CodeBench, Yagarto, Linaro, stb.). Jelen dokumentációban mi a Sourcery CodeBench használatát írjuk le (a többi fordító eszköz telepítése és használata nagyon hasonlóan végezhető).

A Sorcery CodeBench letölthető az alábbi linkről:
http://www.mentor.com/embedded-software/sourcery-tools/sourcery-codebench/editions/lite-edition/arm-eabi

A letöltéshez meg kell adni egy érvényes e-mail címet, ahova a website automatikusan el fogja küldeni a csomag elérhetőségét.
Töltsük le a legújabb verziót, és telepítsük azt a Windows rendszereken megszokott módon.

OpenOCD debugger

Az elkészített és lefordított szoftverünket valamilyen módon ki is kell próbálnunk az eszközön, az esetlegesen elkövetett programozási hibáinkat meg kell találnunk. Ehhez nyújt támogatást a szabad forrású OpenOCD debugger.
Az OpenOCD egy olyan különálló eszköz, mely képes TCP kapcsolaton keresztül kapott parancsokat a számítógéphez csatlakoztatott JTAG vagy SWD debugger eszköznek továbbítani (az STM32F4Discovery kártya SWD debugger áramkört tartalmaz).
Az OpenOCD Windows rendszerekre lefordított verziója elérhető az alábbi linken:
http://www.freddiechopin.info/pl/download/category/4-openocd?download=76%3Aopenocd-0.6.0

Törekedjünk arra, hogy a legújabb verziót használjuk az eszközből. Az eszköz telepítést nem igényel, a letöltött csomagot ki kell tömörítenünk, és használatra kész.

STLink Utility

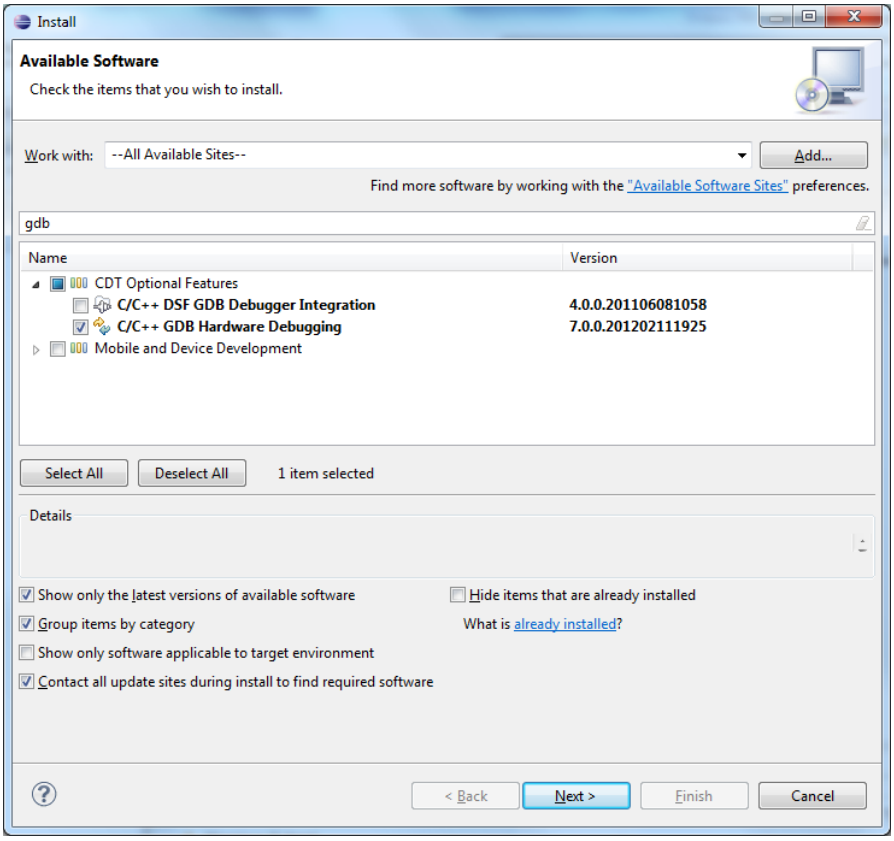
Az STM32F4Discovery kártyán található SWD debugger USB vonalon keresztül csatlakozik a PC-hez. Az STLink Utility tartalmazza egyrészt az eszközmeghajtó szoftvert, másrészt lehetőséget biztosít az eszköz flash memória tartalmának megtekintésére, programozására.

Az STLink Utility letölthető az alábbi linkről:
http://www.st.com/internet/com/SOFTWARE_RESOURCES/TOOL/DEVICE_PROGRAMMER/stm32_st-link_utility.zip

Töltsük le a legújabb verziót, és telepítsük azt a Windows rendszereken megszokott módon.

C/C++ GDB Hardware Debugging modul

Az Eclipse rendszer alap kiépítésében a PC-re történő szoftverfejlesztéshez szükséges modulokat tartalmazza. Nekünk egy távoli eszközt kellene programoznunk, ehhez már rendelkezésünkre áll a fizikai kapcsolatot megvalósító eszköz (OpenOCD), de ezt valamilyen módon az Eclipse rendszerhez is csatlakoztatnunk kell, erre szolgál a C/C++ Hardware Debugging modul.
A modul telepítéséhez indítsuk el az Eclipse rendszert, a Help menüből válasszuk ki az Install New Software menüpontot.
A *Work with* legördülő listából válasszuk ki az *All Available Sites* opciót, és az alatta található szövegablakba írjuk be a gdb kulcsszót. Az Eclipse megkeresi az elérhető csomagokat.
Válasszuk ki a *C/C++ GDB Hardware Debugging* modult a *CDT Optional Features* szekcióból és indítsuk el a telepítését a *Next* gomb megnyomásával.

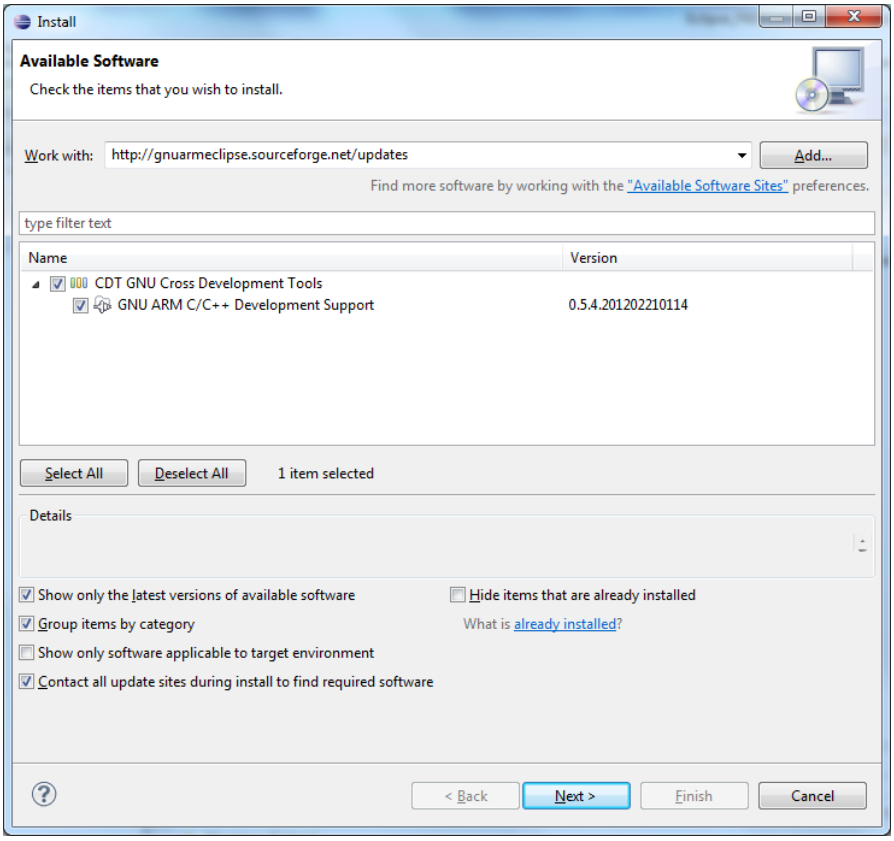


1. ábra C/C++ GDB Hardware Debugging modul telepítése

GNU ARM Eclipse modul

A nyílt forrású gcc fordítók működését makefile segítségével befolyásolhatjuk, ez a makefile tartalmazza a lefordítandó fájlok elérési útjait, a compiler és a linker beállításait, stb. A makefile megírása nem magától értetődő, ismerni kell hozzá a compiler és a linker összes lehetséges beállítását, és a követendő szintaktikát.
Ahhoz, hogy ne kelljen kézzel megírnunk a makefile-t szükségünk van egy Eclipse modulra, ami elvégzi helyettünk ezt a feladatot.
A modul telepítéséhez az Eclipse rendszerben a Help menüből válasszuk ki a már korábban is használt *Install New Software* menüpontot.
A modul nem az Eclipse projekt része, ezért meg kell adnunk azt a címet, ahonnan az eszköz letölthető.
A *Work with* szerkesztőablakba másoljuk be az alábbi címet és nyomjunk *Enter*t.
http://gnuarmeclipse.sourceforge.net/updates

Válasszuk ki és telepítsük a *GNU ARM C/C++ Development Support* modult a *Next* gomb megnyomásával.



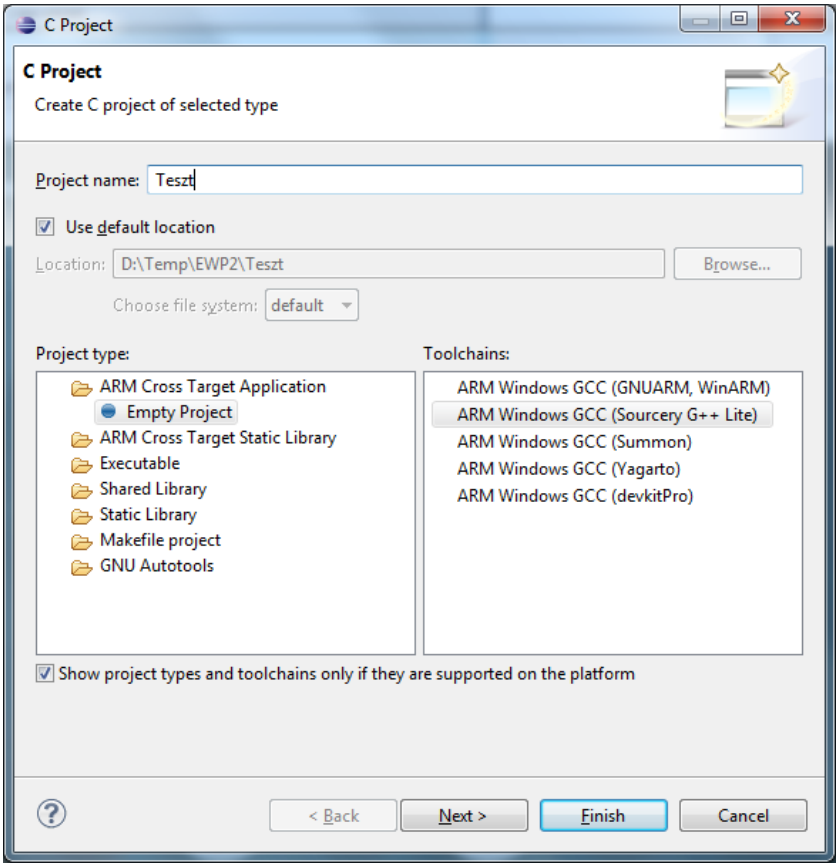
2. ábra A GNU ARM modul telepítése

3. Fejlesztés Eclipse-ben

A korábbi fejezetben leírt telepítési lépések elvégzése után készen áll a fejlesztőkörnyezetünk, már csak szoftvert kellene készíteni az STM32F4Discovery kártyára.

Projekt létrehozása

Az *Eclipse File/New/C Project* menüpontját kiválasztva az alábbi ábrán látható ablak jelenik meg.
Adjunk nevet a projektünknek, majd válasszuk ki az *ARM Cross Target Application* blokkból az *Empty project* opciót.
Válasszuk ki a fordító eszközt - *ARM Windows GCC(Sourcery G++ Lite)*
A *Finish* gomb megnyomásával létrejön a projekt.

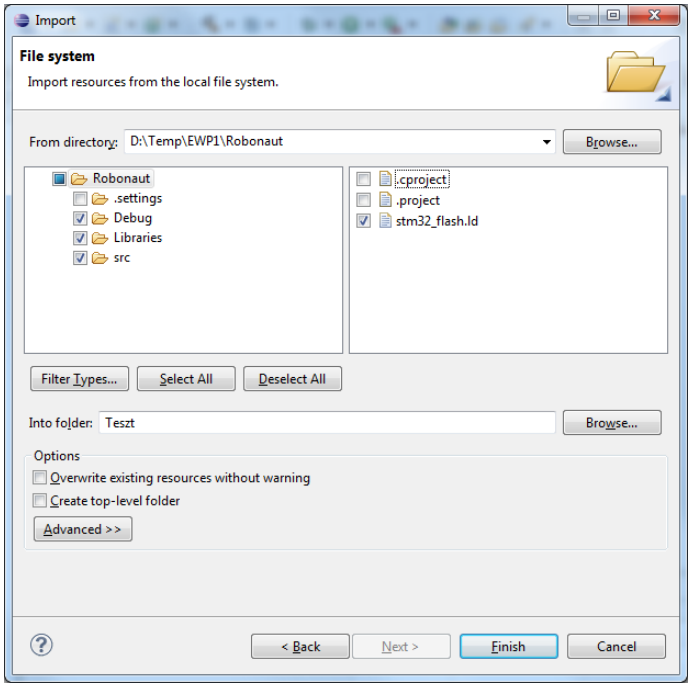


3. ábra Új C projekt létrehozása

Fájlok hozzáadása a projekthez

Leggyakrabban nem teljesen nulláról építjük fel projektünket, hanem valamilyen meglévő alkalmazásvázat használunk kiindulási alapként, és ezt bővítjük az igényeinknek megfelelően.
Az alkalmazásváz importálásához menjünk el a *File/Import…* menübe, ott pedig válasszuk ki a *General/File System* opciót
Az alábbi ablakban adjuk meg az importálandó fájlok elérési útját. (A példában a jelen dokumentum mellé kiadott mintaalkalmazást importáljuk).

Válasszuk ki a *Debug*, a *Libraries* és az *src* könyvtárakat, valamint az *stm32_flash.ld* állományt, majd a *Finish* gombra kattintva importáljuk azokat.

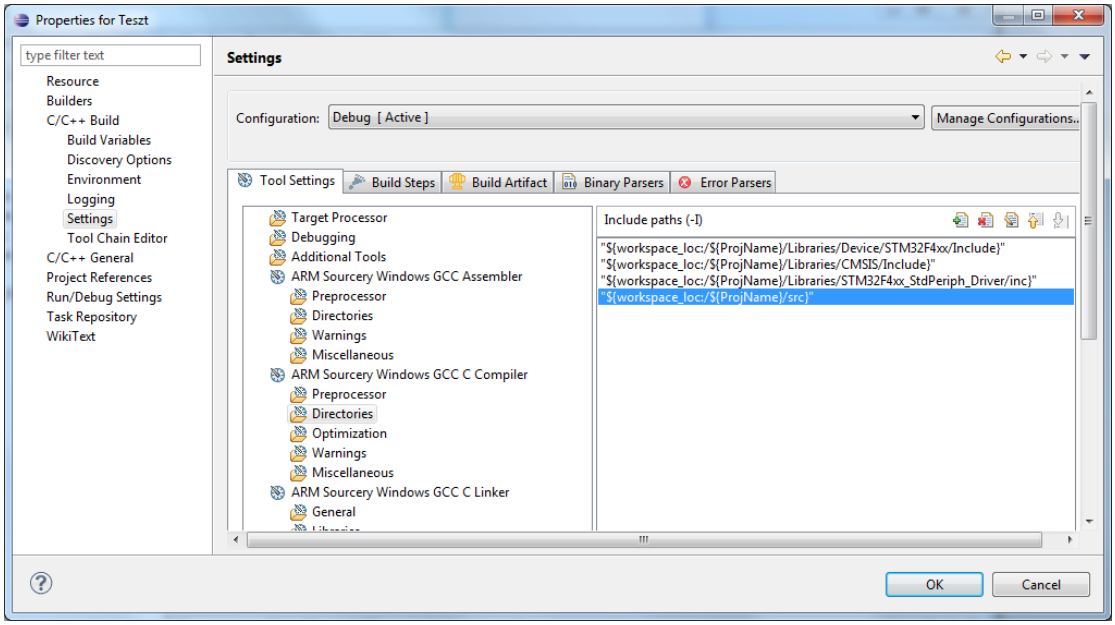


4. ábra Projekt fájljainak importálása

A fordító eszközök beállításai

Ahhoz, hogy a szoftverünk hibák nélkül lefordúljon, be kell állítanunk a fordítókörnyezetet a *Project/Properties/ C/C++ Build/ Settings* ablakban.

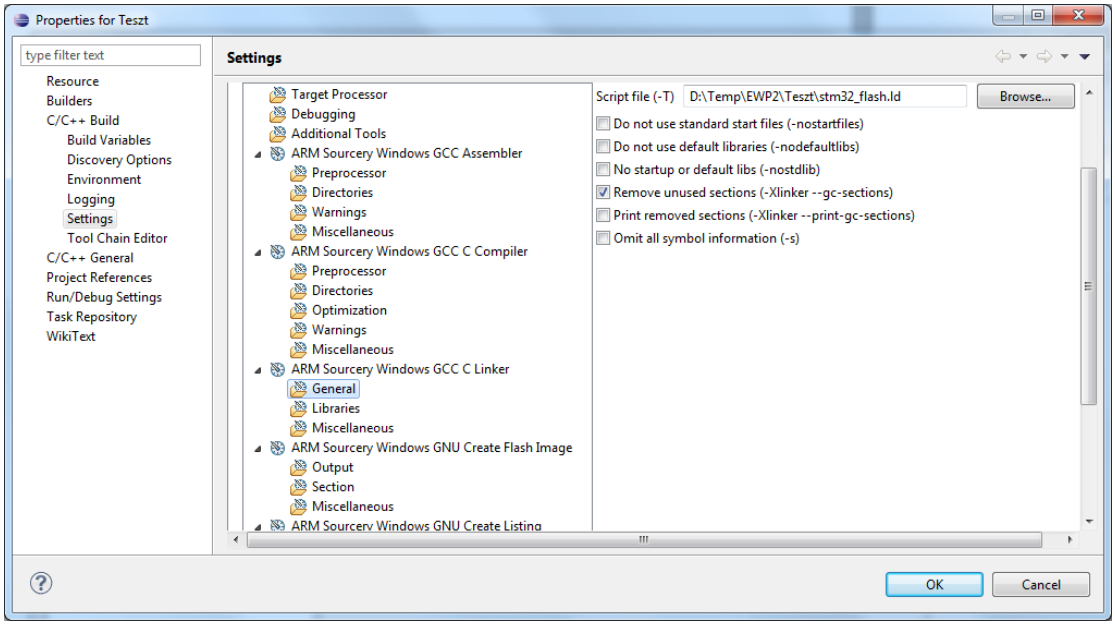
1. Az include könyvtárak beállítása
   A *Compiler Directories* beállításainál állítsuk be azokat a könyvtárakat, ahol a projekthet szükséges header fájlokat tároljuk (a könyvtárakat a Workspace-hez képesti referenciaként állítsuk be, így a projektünk elmozgatása esetén nem kell újra beállítani ezeket)



5. ábra Include könyvtárak beállítása

2. A linker beállításai

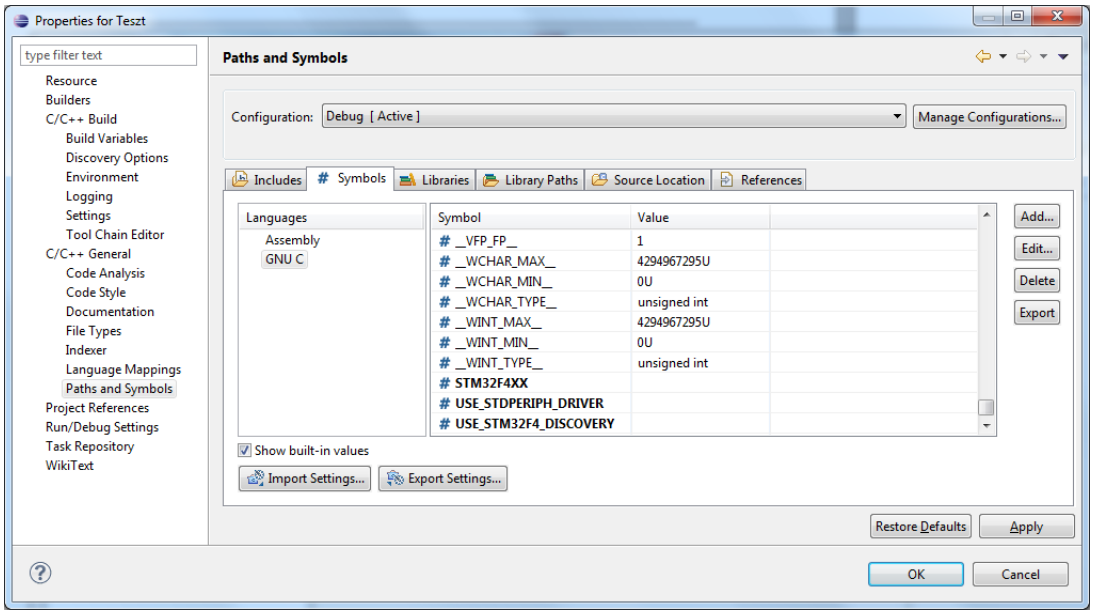
   A *Linker General* beállításainál állítsuk be a *Script Fájl*t (legyen ez a projektünkbe importált *stm32_flash.ld*, ez mondja meg a linkernek, hogy mit hová helyezzen a memóriában), sajnos ezt csak fájlrendszer hivatkozásként lehet beállítani, a workspace relatív hivatkozást nem lehet használni.
   Valamint kapcsoljuk ki a *Do not use standard start files* opciót és kapcsoljuk be a *Remove unused sections* opciót.



6. ábra A linker beállításai

3. Precompiler szimbólumok beállítása

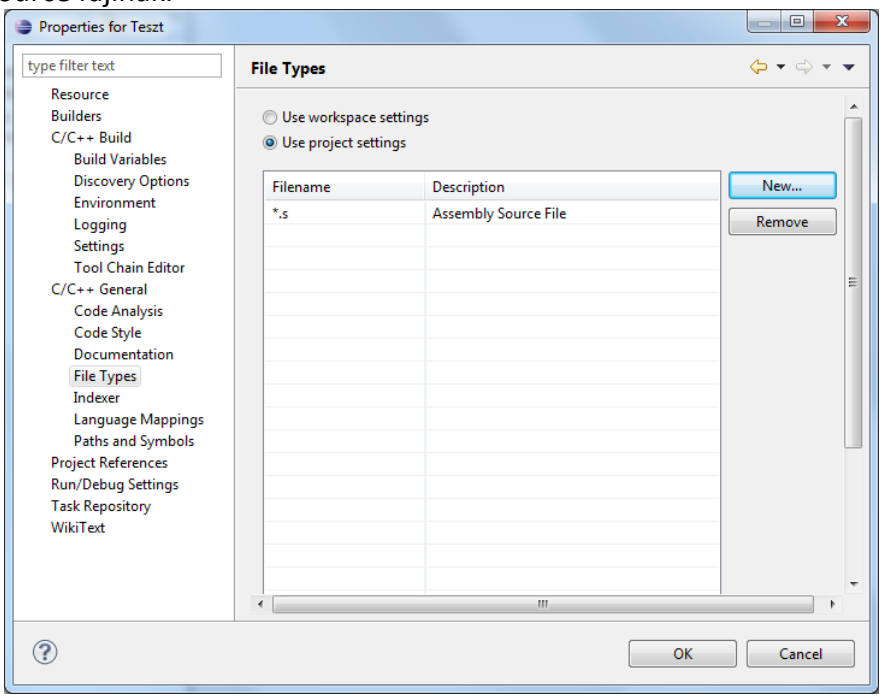
   A *C/C++ General / Path and Symbols / Symbols / GNU C* ablakban adjuk hozzá az alábbi ábrán vastagon szedett szimbólumokat.



7. ábra Precompiler szimbólumok megadása

4. Assembly fájlok kiterjesztésének beállítása

A *C/C++ General / File Types* ablakban állítsuk be a *.s fájlokat assembly source fájlnak.

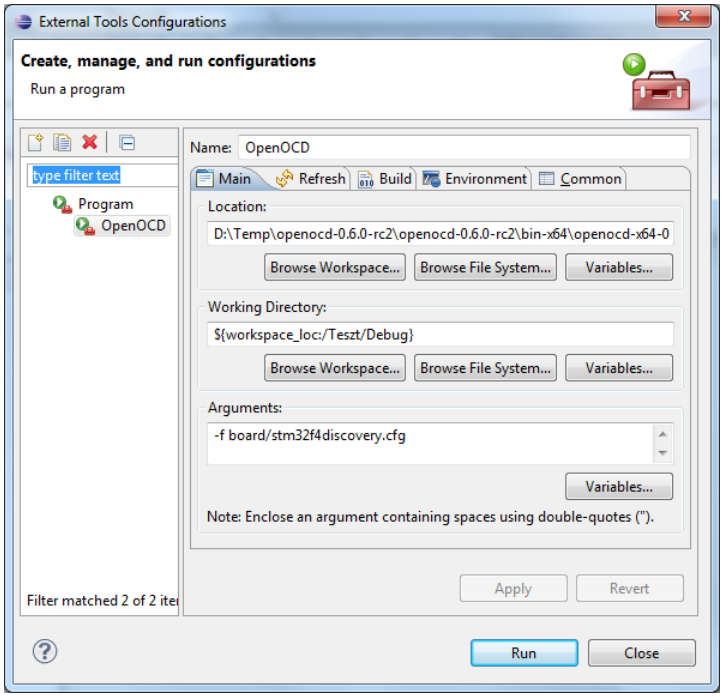


8. ábra Assembly fájlok kiterjesztésének beállítása

Fordítsuk le az alkalmazást!

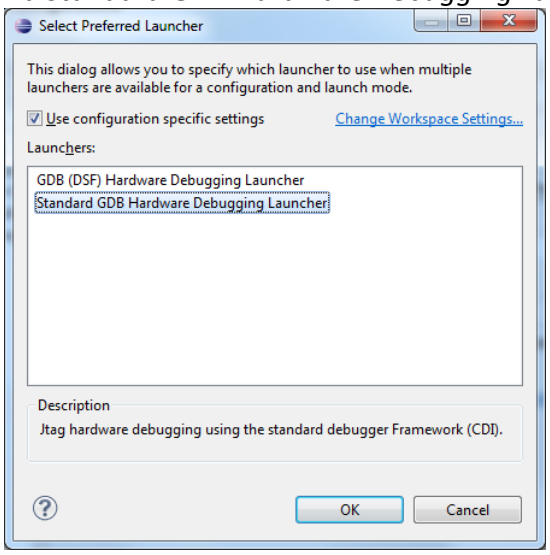
A debugger beállításai

1. <u>OpenOCD beállítása debuggernek</u>
   - Az Eclipse eszköztáron keressük meg az *External Tools* gombot, nyissuk le, és lépjünk be az *External Tools Configurations* ablakba.
   - Válasszuk ki a *Program* opciót, és kattintsunk a *New Launch Configuration* gombra.
   - Adjunk nevet a konfigurációnknak – legyen OpenOCD
   - Állítsuk be az OpenOCD elérési útját (ahová kitömörítettük a csomagot), keressük meg az OpenOCD… exe fájlt (32 bites rendszereken a bin könyvtárban, 64 bites rendszereken a bin-x64 könyvtárban)
   - A *Working directory*-hoz állítsuk be a projektünk Debug könyvtárát.
   - Az argumentumokhoz állítsuk be a *–f board/stm32f4discovery.cfg* opciót.
   - A *Common* fülön állítsuk be, hogy a konfigurációnk megjelenjen a Favorites menüben
   - Mentsük a beállításokat az *Apply* gombra kattintással.
   - Kattintsunk a *Run* gombra, ekkor elindul az OpenOCD (ha minden rendben, akkor a konzolon néhány warning kíséretében megmondja, hogy hány hardveres break- és watchpointot támogat az eszköz)



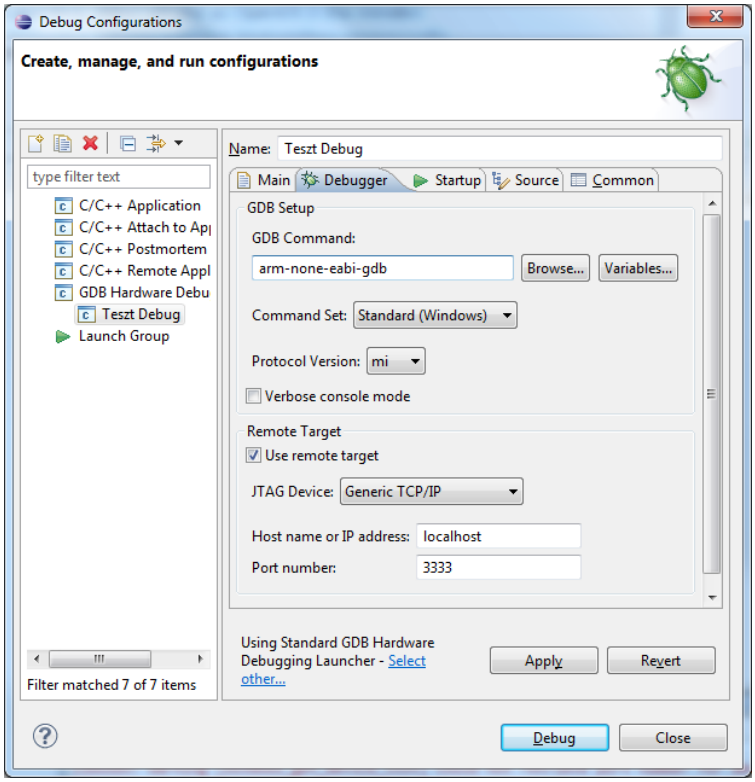
9. ábra OpenOCD beállítása

2. <u>Debug konfiguráció összeállítása</u>
   - Az Eclipse eszköztáron keressük meg a Debug gombot, nyissuk le, és lépjünk be a *Debug Configurations* ablakba.
   - Válasszuk ki a *GDB Hardware Debugging* opciót, és kattintsunk a *New Debug Configuration* gombra.
   - A *Main* fülön az ablak alján a *Select Other…* linkre kattintusnk rá, és válasszuk ki a *Standard GDB Hardware Debugging Launcher* opciót



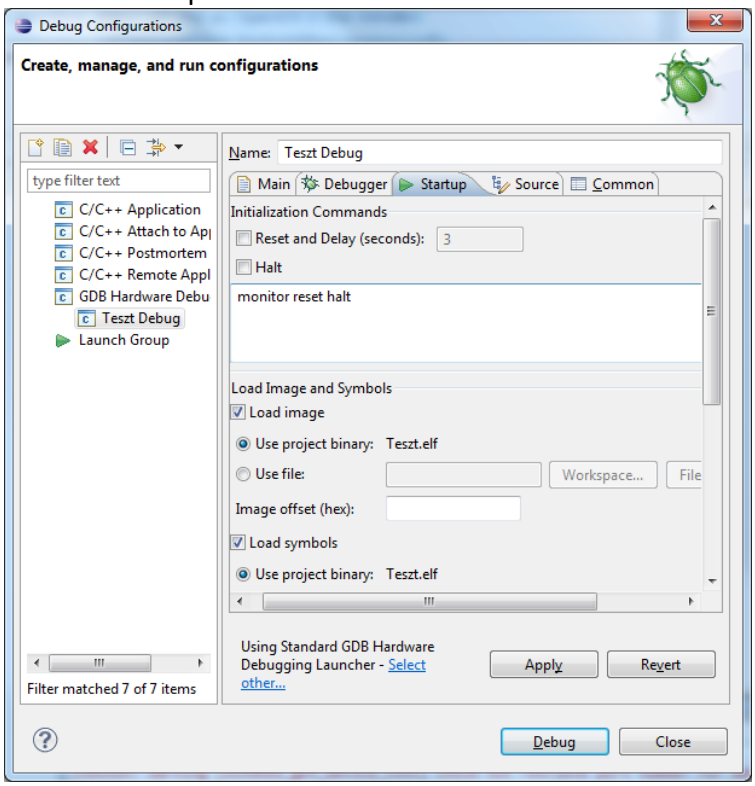
10. ábra Standard Launcher kiválasztása

   - A *Debugger* fülön a *GDB Command* mező értékét állítsuk arm-none-eabi-gdb értékre, a *Port number* mező értékét pedig 3333-ra.



11. ábra Debugger beállítása

- A Startup fülön a *Reset and Delay seconds* opciót kapcsoljuk ki, illetve a *Halt opciót* is kapcsoljuk ki. A Halt alatti szerkesztőmezőbe írjuk be a *monitor reset halt* parancsokat.

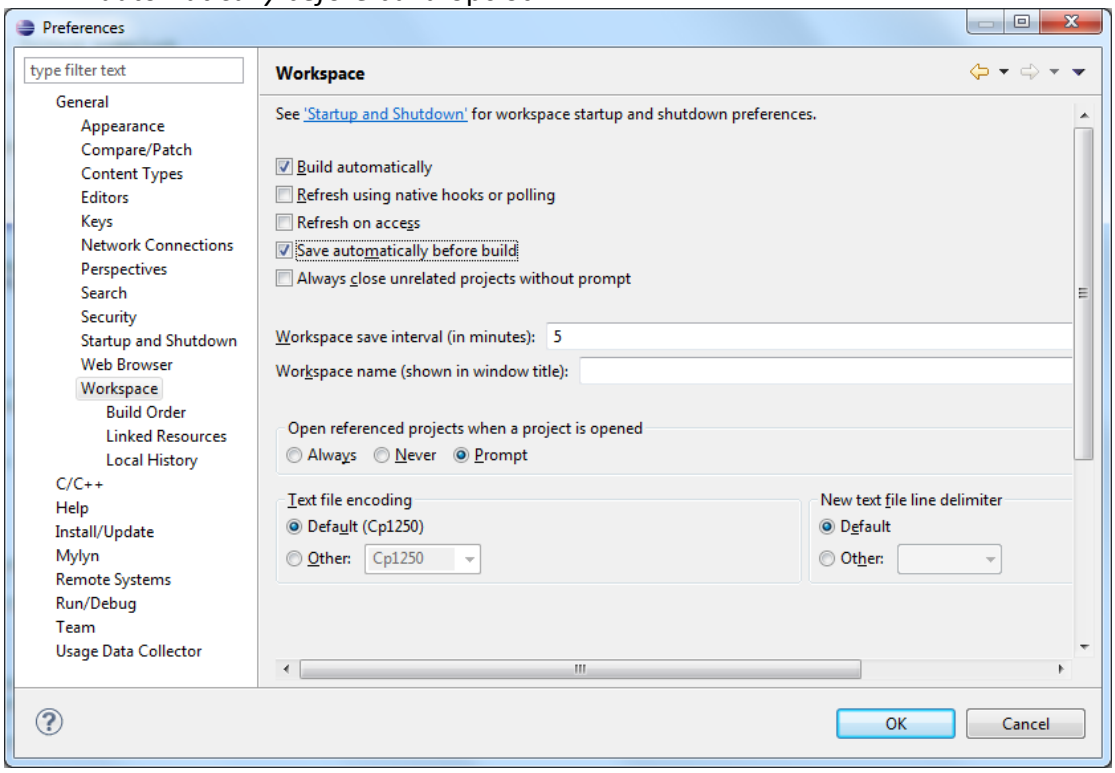


12. ábra Debugger startup beállítása

- A *Common* fülön állítsuk be, hogy a konfigurációnk megjelenjen a Favorites menüben
- Mentsük a beállításokat az *Apply* gombra történő kattintással
- A továbbiakban debugolni úgy lehetséges, hogy az eszköztáron a *Debug* gombon kiválasztjuk a most elkészített konfigurációt, ilyenkor a Debugger elindul, és átkapcsolja a felületet Debug nézetre (ha nem megy a debugolás, akkor győződjünk meg róla, hogy az OpenOCD egyetlen egy példányban fut-e).

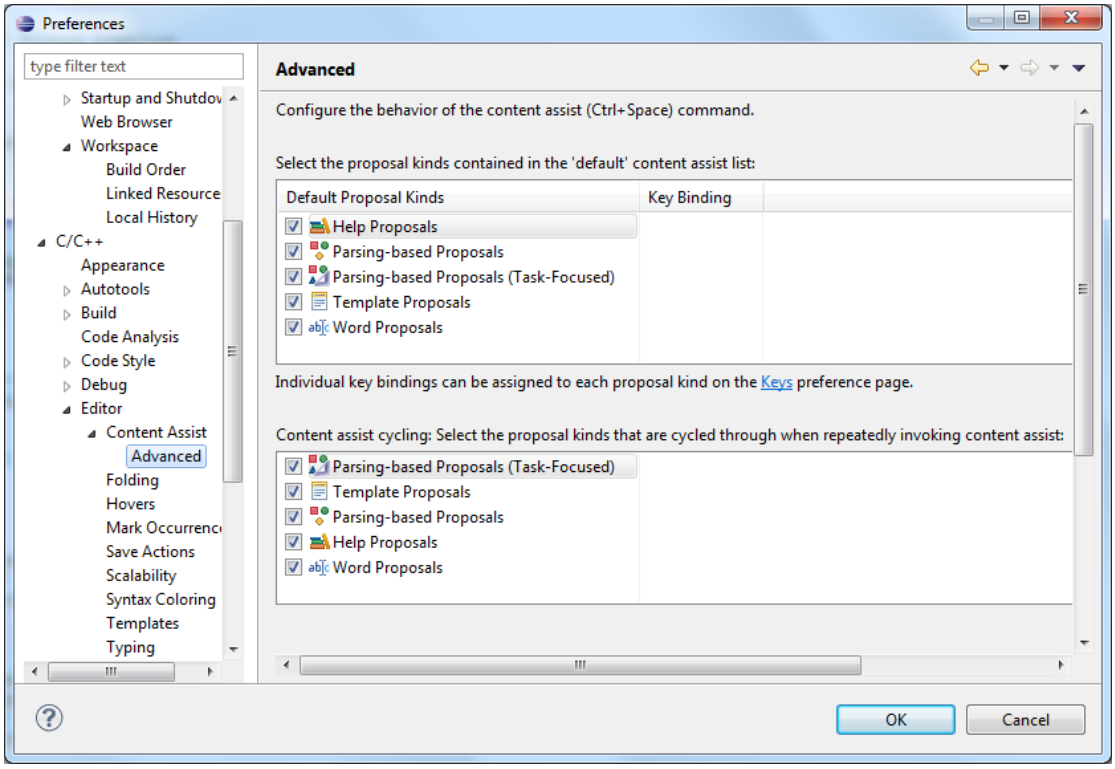
Egyéb hasznos beállítások

1. Automatikus mentés Build előtt
   Az Eclipse-ben *Window/Preferences* menüpontot válasszuk ki, a megjelenő ablakban navigáljunk el a *General/Workspace* oldalra, és kapcsoljuk be a *Save automatically before build* opciót.



13. ábra Automatikus mentés bekapcsolása

2. Intellisense beállítások
   Az Eclipse-ben *Window/Preferences* menüpontot válasszuk ki, a megjelenő ablakban navigáljunk el a *C/C++/Editor/Content Assist/Advanced* oldalra, és kapcsoljuk be az összes opciót.
   Ahhoz, hogy az aktuális projektünkre érvényesek legyenek a beállítások, a Project explorerben a projekt nevén jobb gombbal kattintva a felugró menüből válasszuk ki az *Index/Rebuild* opciót.



14. ábra IntelliSense beállítások

Amennyiben a fent leírt lépéseket megfelelően elvégeztük, a fejlesztőkörnyezet készen áll arra, hogy elkészítsük, debugoljuk alkalmazásainkat.

Sok sikert a fejlesztéshez!