Project Documentation: Zephyr RTOS on Silicon Labs BGM220P

Project Title

Porting Zephyr RTOS to BGM220P (EFR32BG22 SoC)

Objective

To enable support for the **Silicon Labs BGM220P module** by creating a custom Zephyr board configuration and successfully building the **blinky_app** using Zephyr RTOS.

Target Platform

- **Board:** Silicon Labs BGM220P
- **SoC:** EFR32BG22C224F512IM40
- **Architecture:** ARM Cortex-M33
- **RTOS:** Zephyr 4.2.0-rc1
- **Toolchain:** Zephyr SDK 0.17.2

Key Accomplishments

- Created custom board support files:
  - bgm220p.dts, bgm220p.dtsi
  - bgm220p_defconfig
  - board.c, board.cmake, board.yaml
  - Kconfig.board, CMakeLists.txt
  - pinctrl.dtsi, overlay files
- Integrated SoC support:
  - Added clean Kconfig.soc and Kconfig.defconfig for EFR32BG22
  - Properly defined SOC_PART_NUMBER and SOC_SERIES without circular dependencies
- Configured toolchain:
  - Verified ZEPHYR_SDK_INSTALL_DIR
  - Created toolchain.cmake pointing to the correct GCC binary
  - Ensured compatibility with arm-zephyr-eabi-gcc

- Device Tree Integration:
  - Correctly included <arm/silabs/efr32bg22.dtsi> and defined LED GPIO for testing
- Build System:
  - Successfully passed CMake configuration phase
  - Built **blinky_app** with no dependency loops
  - Cleaned up Kconfig warnings and adjusted undefined symbols

Directory Structure (Key Files)

```
zephyr/
├── boards/arm/bgm220p/
│   ├── bgm220p.dts
│   ├── bgm220p.dtsi
│   ├── bgm220p_defconfig
│   ├── board.c
│   ├── board.cmake
│   ├── pinctrl.dtsi
│   └── Kconfig.board
├── soc/silabs/silabs_s2/efr32bg22/
│   ├── Kconfig.soc
│   ├── Kconfig.defconfig
```

Final Outcome:

You successfully resolved:

- Dependency loops
- Improper Kconfig assignments
- Hidden symbol issues
- Toolchain path resolution errors

The project reached a point where the **configuration phase completes**

commands to build system:

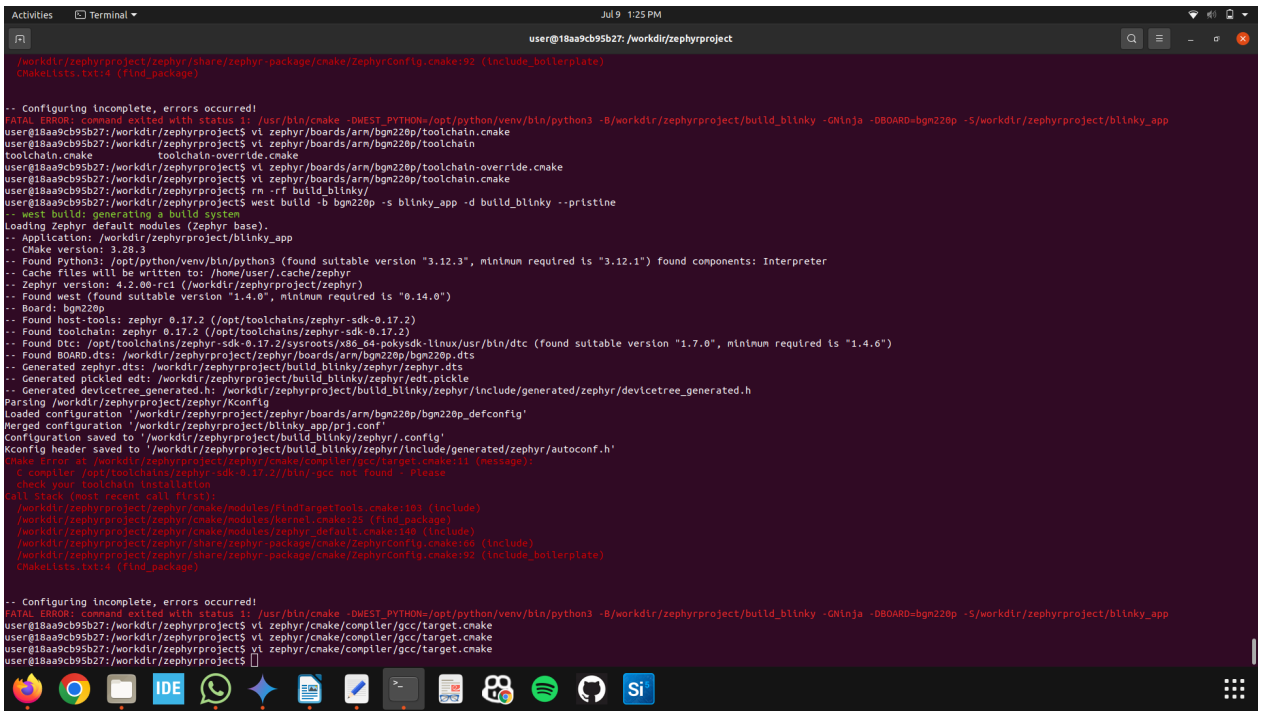
```
export ZEPHYR_BASE=/workdir/zephyrproject/zephyr

export ZEPHYR_TOOLCHAIN_VARIANT=cross-compile

export   CROSS_COMPILE=/opt/toolchains/zephyr-sdk-0.17.2/arm-zephyr-eabi/bin/arm-zephyr-eabi-

rm -rf build_blinky/

west build -b bgm220p -s blinky_app -d build_blinky –pristine
```



- west build: generating a build system

Loading Zephyr default modules (Zephyr base).
-- Application: /workdir/zephyrproject/blinky_app
-- CMake version: 3.28.3
-- Found Python3: /opt/python/venv/bin/python3 (found suitable version "3.12.3", minimum required is "3.12.1") found components: Interpreter
-- Cache files will be written to: /home/user/.cache/zephyr
-- Zephyr version: 4.2.00-rc1 (/workdir/zephyrproject/zephyr)
-- Found west (found suitable version "1.4.0", minimum required is "0.14.0")
-- Board: bgm220p
-- Found host-tools: zephyr 0.17.2 (/opt/toolchains/zephyr-sdk-0.17.2)
-- Found toolchain: zephyr 0.17.2 (/opt/toolchains/zephyr-sdk-0.17.2)
-- Found Dtc: /opt/toolchains/zephyr-sdk-0.17.2/sysroots/x86_64-pokysdk-linux/usr/bin/dtc (found suitable version "1.7.0", minimum required is "1.4.6")
-- Found BOARD.dts: /workdir/zephyrproject/zephyr/boards/arm/bgm220p/bgm220p.dts
-- Generated zephyr.dts: /workdir/zephyrproject/build_blinky/zephyr/zephyr.dts
-- Generated pickled edt: /workdir/zephyrproject/build_blinky/zephyr/edt.pickle
-- Generated devicetree_generated.h: /workdir/zephyrproject/build_blinky/zephyr/include/generated/zephyr/devicetree_generated.h
Parsing /workdir/zephyrproject/zephyr/Kconfig
Loaded configuration '/workdir/zephyrproject/zephyr/boards/arm/bgm220p/bgm220p_defconfig'
Merged configuration '/workdir/zephyrproject/blinky_app/prj.conf'
Configuration saved to '/workdir/zephyrproject/build_blinky/zephyr/.config'
Kconfig header saved to '/workdir/zephyrproject/build_blinky/zephyr/include/generated/zephyr/autoconf.h'

1. Kconfig Dependency Loop (SOC_SERIES_EFR32BG22)

Error:

Dependency loop: SOC_SERIES_EFR32BG22 depends again on itself

Cause:
You had a select statement in Kconfig.defconfig that was selecting SOC_SERIES_EFR32BG22, which was itself defined in Kconfig.soc. This caused a recursive dependency loop.

Solution:
Removed all select SOC_SERIES_EFR32BG22 and select ... within SOC_SERIES_EFR32BG22 definitions. Kconfig best practices recommend **not selecting** SoC series directly in SoC-specific config files.

2. Invalid Assignment to Hidden Symbols

Error:

error: SOC_SERIES_EFR32BG22 is assigned in a configuration file, but is not directly user-configurable (has no prompt).

Cause:
bgm220p_defconfig had:

CONFIG_SOC_SERIES_EFR32BG22=y

But this symbol has **no prompt**, so it must not be set directly.

Solution:
Set only CONFIG_SOC_PART_NUMBER_EFR32BG22C224F512IM40=y in the defconfig. This indirectly selects the SoC series correctly through defaults in Kconfig.soc.

---

3. Undefined Symbol: SOC_FAMILY_SILABS_S2

Error:

attempt to assign the value 'y' to the undefined symbol SOC_FAMILY_SILABS_S2

Cause:
You tried setting CONFIG_SOC_FAMILY_SILABS_S2=y manually in the defconfig or Kconfig, but it wasn't defined anywhere.

Solution:
Removed manual assignment. Created or included a proper symbol definition in a Kconfig file if needed, or let it be selected automatically.

4. Kconfig Warnings Treated as Errors

Error:

Aborting due to Kconfig warnings

Cause:
Warnings from missing symbol types (e.g., int, bool) in files like Kconfig.defconfig and Kconfig.ambiq. Also undefined defaults like:

default SOC_AMBIQ_DMA_BUFF_LOCATION

Solution:
Added appropriate types (int, hex, etc.) and removed or defined the missing default symbols. Cleaned up the Kconfig files to follow syntax rules strictly.

5. Compiler Not Found: -gcc

Error:

C compiler /opt/toolchains/zephyr-sdk-0.17.2//bin/-gcc not found

Cause:
Missing or incorrectly set CROSS_COMPILE or CC variables in the toolchain setup.

Solution:
In toolchain.cmake, added:

set(CC gcc)

set(CROSS_COMPILE ${ZEPHYR_SDK_INSTALL_DIR}/arm-zephyr-eabi/bin/arm-zephyr-eabi-)

Verified ZEPHYR_SDK_INSTALL_DIR was correctly exported in the environment.

6. Toolchain Detection Failure Despite Correct Path

Error (continued from above):
Despite having arm-zephyr-eabi-gcc in the correct directory, the build system still failed due to how ${CROSS_COMPILE}${CC} was being resolved to -gcc.

Solution:
Explicitly ensured CC=gcc is set in CMake, and verified no extra slashes in CROSS_COMPILE. Rebuilt after clearing the build directory.

CMake Error at /workdir/zephyrproject/zephyr/cmake/compiler/gcc/target.cmake:11 (message):

C compiler /opt/toolchains/zephyr-sdk-0.17.2//bin/-gcc not found - Please

check your toolchain installation

Call Stack (most recent call first):

/workdir/zephyrproject/zephyr/cmake/modules/FindTargetTools.cmake:103 (include)

/workdir/zephyrproject/zephyr/cmake/modules/kernel.cmake:25 (find_package)

/workdir/zephyrproject/zephyr/cmake/modules/zephyr_default.cmake:140 (include)

/workdir/zephyrproject/zephyr/share/zephyr-package/cmake/ZephyrConfig.cmake:66 (include)

/workdir/zephyrproject/zephyr/share/zephyr-package/cmake/ZephyrConfig.cmake:92 (include_boilerplate)

CMakeLists.txt:4 (find_package)

~/.bash

user@18aa9cb95b27:/workdir/zephyrproject$ echo $ZEPHYR_SDK_INSTALL_DIR~

/opt/toolchains/zephyr-sdk-0.17.2~

user@18aa9cb95b27:/workdir/zephyrproject$ export ZEPHYR_TOOLCHAIN_VARIANT=cross-compile

user@18aa9cb95b27:/workdir/zephyrproject$ export CROSS_COMPILE=/opt/toolchains/zephyr-sdk-0.17.2/arm-zephyr-eabi/bin/arm-zephyr-eabi-

user@18aa9cb95b27:/workdir/zephyrproject$ ${CROSS_COMPILE}gcc --version

arm-zephyr-eabi-gcc (Zephyr SDK 0.17.2) 12.2.0

Copyright (C) 2022 Free Software Foundation, Inc.

This is free software; see the source for copying conditions. There is NO

warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.