

目次

ビルトインアプリケーションの追加	3
準備	3
作業	3
ディレクトリ、ファイルの追加	3
config の変更	6
ビルド	11
実機への書き込みと実行	11

ビルトインアプリケーションの追加

[ファームウェアの書き込みと動作](#) までで NuttX のシェル(NuttShell) が起動するところまで確認できました。

NuttX は便利なシェルが利用できますので、アプリケーションの開発方法としては、

- アプリケーションは各機能を **Task/Thread** に小分けにして、**プロセス間通信¹⁾** で全体の機能を実現
- 各機能を実装したタスクを **NuttShellから起動** するビルトインアプリケーションを実装
- 起動時に NuttShell から各タスクを自動起動する (/etc/init.d/rcS から)²⁾

というふうを実装するとデバッグが容易になります。

そこで、まずはビルトインアプリケーションをどのように作るか、について記載します。

マニュアルは <https://cwiki.apache.org/confluence/display/NUTTX/Custom+Application+Directories> にあります。

準備

NuttX のソースを clone した2つのディレクトリのうち、アプリケーションは **apps/** ディレクトリ以下に作成します。

編集したあとでも元に戻せるよう、新しい作業用ブランチを作成します。

```
develop:~/src/NuttX_XG50$ cd apps/  
develop:~/src/NuttX_XG50/apps$ git checkout -b hello_test  
Switched to a new branch 'hello_test'
```

作業

ディレクトリ、ファイルの追加

apps/ ディレクトリ以下は、下のようになっています。

```
develop:~/src/NuttX_XG50/apps$ ls -lnF  
total 260  
-rw-rw-r-- 1 1000 1000 4166 3月 7 09:15 Application.mk  
drwxrwxr-x 14 1000 1000 4096 3月 8 14:53 BearSSL/  
-rw-rw-r-- 1 1000 1000 8793 3月 7 09:15 COPYING  
-rw-rw-r-- 1 1000 1000 145150 3月 7 09:17 ChangeLog.txt  
-rw-rw-r-- 1 1000 1000 2797 3月 7 09:15 Directory.mk  
-rw-rw-r-- 1 1000 1000 3095 3月 7 09:15 Make.defs  
-rw-rw-r-- 1 1000 1000 5297 3月 7 09:15 Makefile
```

```
-rw-rw-r-- 1 1000 1000 9111 3月 7 09:15 README.txt
drwxrwxr-x 3 1000 1000 4096 3月 8 14:56 builtin/
drwxrwxr-x 6 1000 1000 4096 3月 8 14:49 canutils/
drwxrwxr-x 121 1000 1000 4096 3月 8 14:49 examples/
drwxrwxr-x 7 1000 1000 4096 3月 8 14:49 fsutils/
drwxrwxr-x 3 1000 1000 4096 3月 8 14:49 gpsutils/
drwxrwxr-x 9 1000 1000 4096 3月 8 14:49 graphics/
drwxrwxr-x 3 1000 1000 4096 3月 7 09:15 import/
drwxrwxr-x 14 1000 1000 4096 3月 7 09:17 include/
drwxrwxr-x 7 1000 1000 4096 3月 8 14:49 interpreters/
drwxrwxr-x 7 1000 1000 4096 3月 7 09:15 modbus/
drwxrwxr-x 23 1000 1000 4096 3月 8 14:49 netutils/
drwxrwxr-x 2 1000 1000 4096 3月 8 14:56 nshlib/
drwxrwxr-x 6 1000 1000 4096 3月 8 14:56 platform/
drwxrwxr-x 34 1000 1000 4096 3月 8 14:49 system/
drwxrwxr-x 2 1000 1000 4096 3月 7 09:15 tools/
drwxrwxr-x 5 1000 1000 4096 3月 8 14:49 wireless/
develop:~/src/NuttX_XG50/apps$
```

テストなので、適当に **test** というディレクトリを作り作業を進めます。

```
develop:~/src/NuttX_XG50/apps$ ls -lnF
total 264
-rw-rw-r-- 1 1000 1000 4166 3月 7 09:15 Application.mk
drwxrwxr-x 14 1000 1000 4096 3月 8 14:53 BearSSL/
-rw-rw-r-- 1 1000 1000 8793 3月 7 09:15 COPYING
-rw-rw-r-- 1 1000 1000 145150 3月 7 09:17 ChangeLog.txt
-rw-rw-r-- 1 1000 1000 2797 3月 7 09:15 Directory.mk
-rw-rw-r-- 1 1000 1000 3095 3月 7 09:15 Make.defs
-rw-rw-r-- 1 1000 1000 5297 3月 7 09:15 Makefile
-rw-rw-r-- 1 1000 1000 9111 3月 7 09:15 README.txt
drwxrwxr-x 3 1000 1000 4096 3月 8 14:56 builtin/
drwxrwxr-x 6 1000 1000 4096 3月 8 14:49 canutils/
drwxrwxr-x 121 1000 1000 4096 3月 8 14:49 examples/
drwxrwxr-x 7 1000 1000 4096 3月 8 14:49 fsutils/
drwxrwxr-x 3 1000 1000 4096 3月 8 14:49 gpsutils/
drwxrwxr-x 9 1000 1000 4096 3月 8 14:49 graphics/
drwxrwxr-x 3 1000 1000 4096 3月 7 09:15 import/
drwxrwxr-x 14 1000 1000 4096 3月 7 09:17 include/
drwxrwxr-x 7 1000 1000 4096 3月 8 14:49 interpreters/
drwxrwxr-x 7 1000 1000 4096 3月 7 09:15 modbus/
drwxrwxr-x 23 1000 1000 4096 3月 8 14:49 netutils/
drwxrwxr-x 2 1000 1000 4096 3月 8 14:56 nshlib/
drwxrwxr-x 6 1000 1000 4096 3月 8 14:56 platform/
drwxrwxr-x 34 1000 1000 4096 3月 8 14:49 system/
drwxrwxr-x 2 1000 1000 4096 3月 8 15:00 test/ <-----
drwxrwxr-x 2 1000 1000 4096 3月 7 09:15 tools/
```

```
drwxrwxr-x  5 1000 1000  4096  3月  8 14:49 wireless/  
develop:~/src/NuttX_XG50/apps$
```

作成した **test** ディレクトリ以下に、

- Kconfig
- Make.defs
- Makefile
- hello_main.c (プログラムソース)

を作成します。

Kconfig

```
#  
# For a description of the syntax of this configuration file,  
# see the file kconfig-language.txt in the NuttX tools repository.  
#  
  
config APP_HELLO  
    bool "\"Hello, World!\" example"  
    default n  
    ---help---  
        Enable the "\"Hello, World!\" example  
  
if APP_HELLO  
  
config APP_HELLO_PROGNAME  
    string "Program name"  
    default "hello"  
    depends on BUILD_KERNEL  
    ---help---  
        This is the name of the program that will be use when the NSH  
ELF  
        program is installed.  
  
config APP_HELLO_PRIORITY  
    int "Hello task priority"  
    default 100  
  
config APP_HELLO_STACKSIZE  
    int "Hello stack size"  
    default 2048  
  
endif
```

Make.defs

```
ifeq ($(CONFIG_APP_HELLO),y)
```

```
CONFIGURED_APPS += test
endif
```

Makefile

```
-include $(TOPDIR)/Make.defs

# Hello, World! built-in application info

CONFIG_APP_HELLO_PRIORITY ?= SCHED_PRIORITY_DEFAULT
CONFIG_APP_HELLO_STACKSIZE ?= 2048

APPNAME = hello
PRIORITY = $(CONFIG_APP_HELLO_PRIORITY)
STACKSIZE = $(CONFIG_APP_HELLO_STACKSIZE)

# Hello, World! Example

ASRCS =
CSRCS =
MAINSRC = hello_main.c

CONFIG_APP_HELLO_PROGNAME ?= hello$(EXEEXT)
PROGNAME = $(CONFIG_APP_HELLO_PROGNAME)

include $(APPDIR)/Application.mk
```

```
<sxh c toolbar:false; title:hello_main.c> #include <nutt/config.h> #include <stdio.h>
```

```
int hello_main(int argc, char *argv[]) {
```

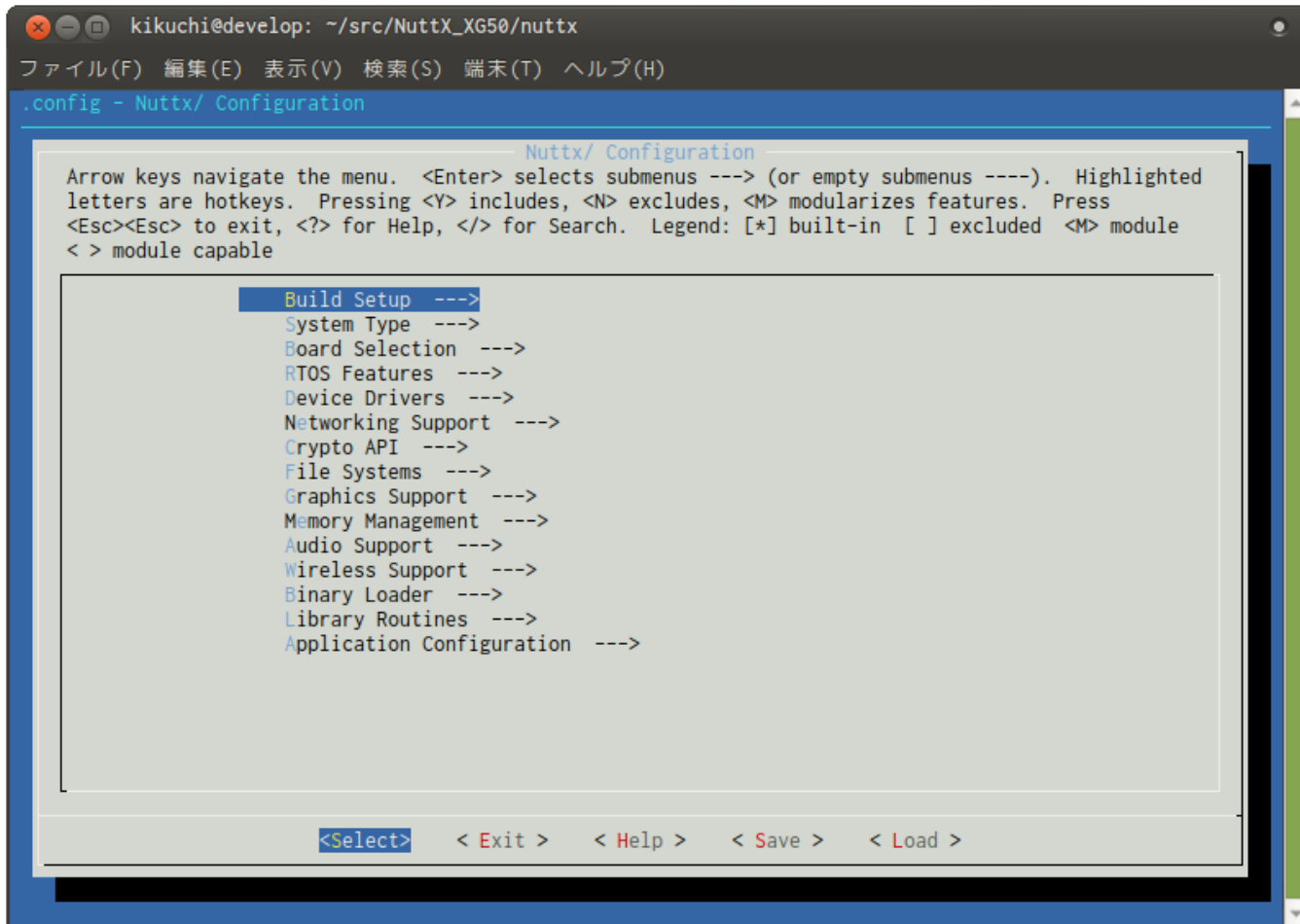
```
printf("Hello, World!!\n");
return 0;
```

```
}
```

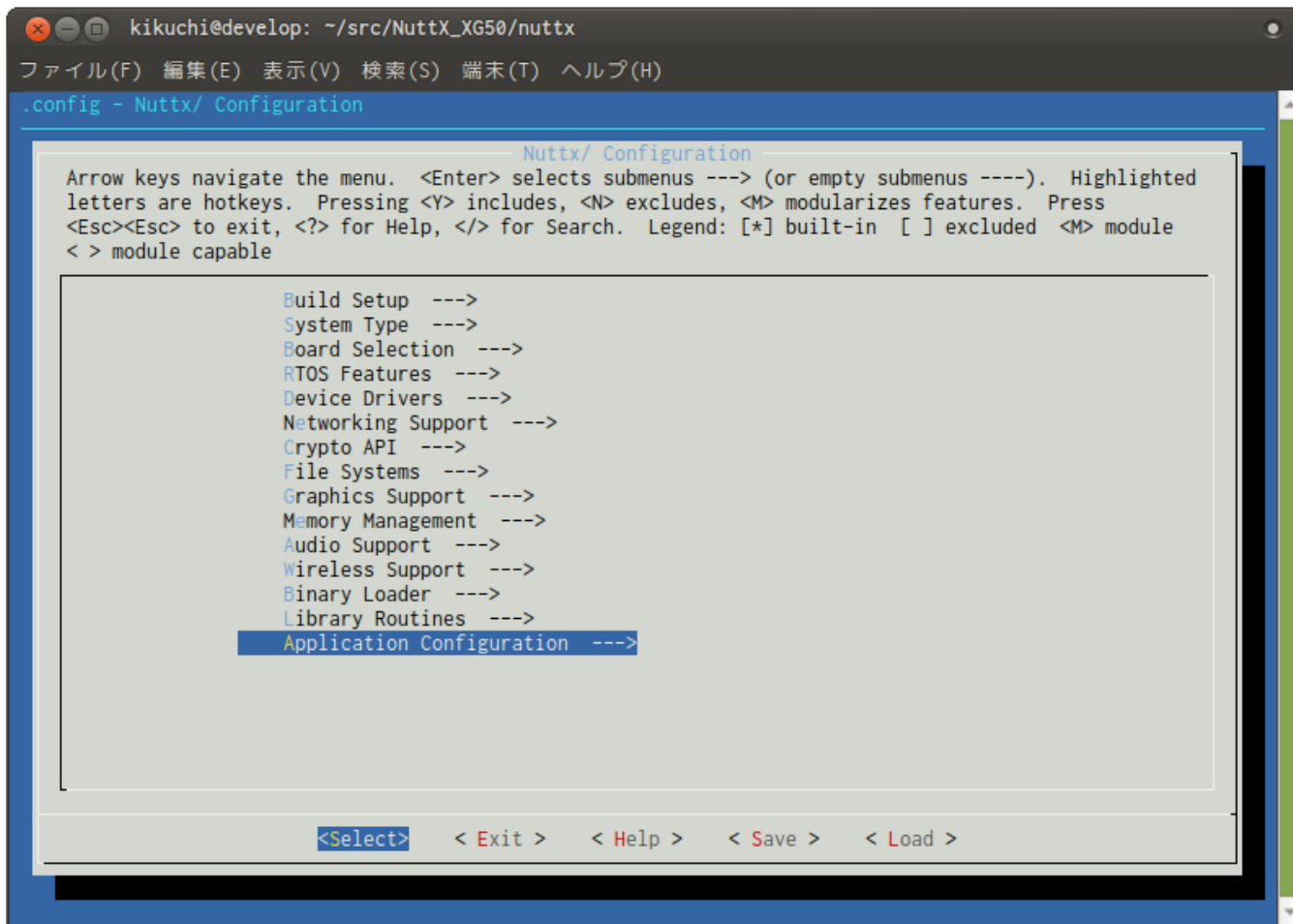
```
</sxh>
```

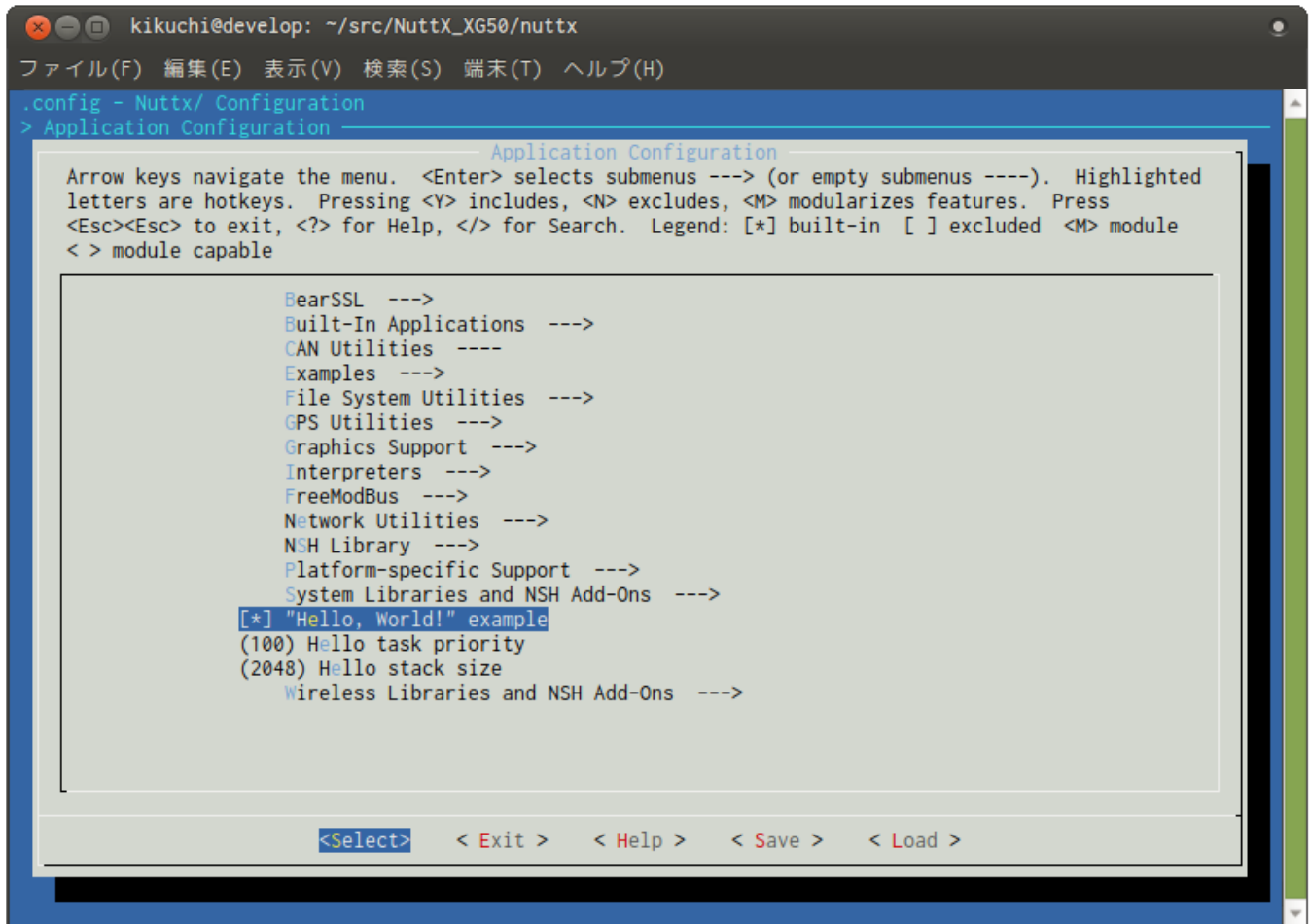
config の変更

nuttx/ ディレクトリで `make clean` した後で `make menuconfig` を実行します。
Linux Kernel と同じような画面になります。



Application Configuration から **“Hello, World!” example** を選択します。



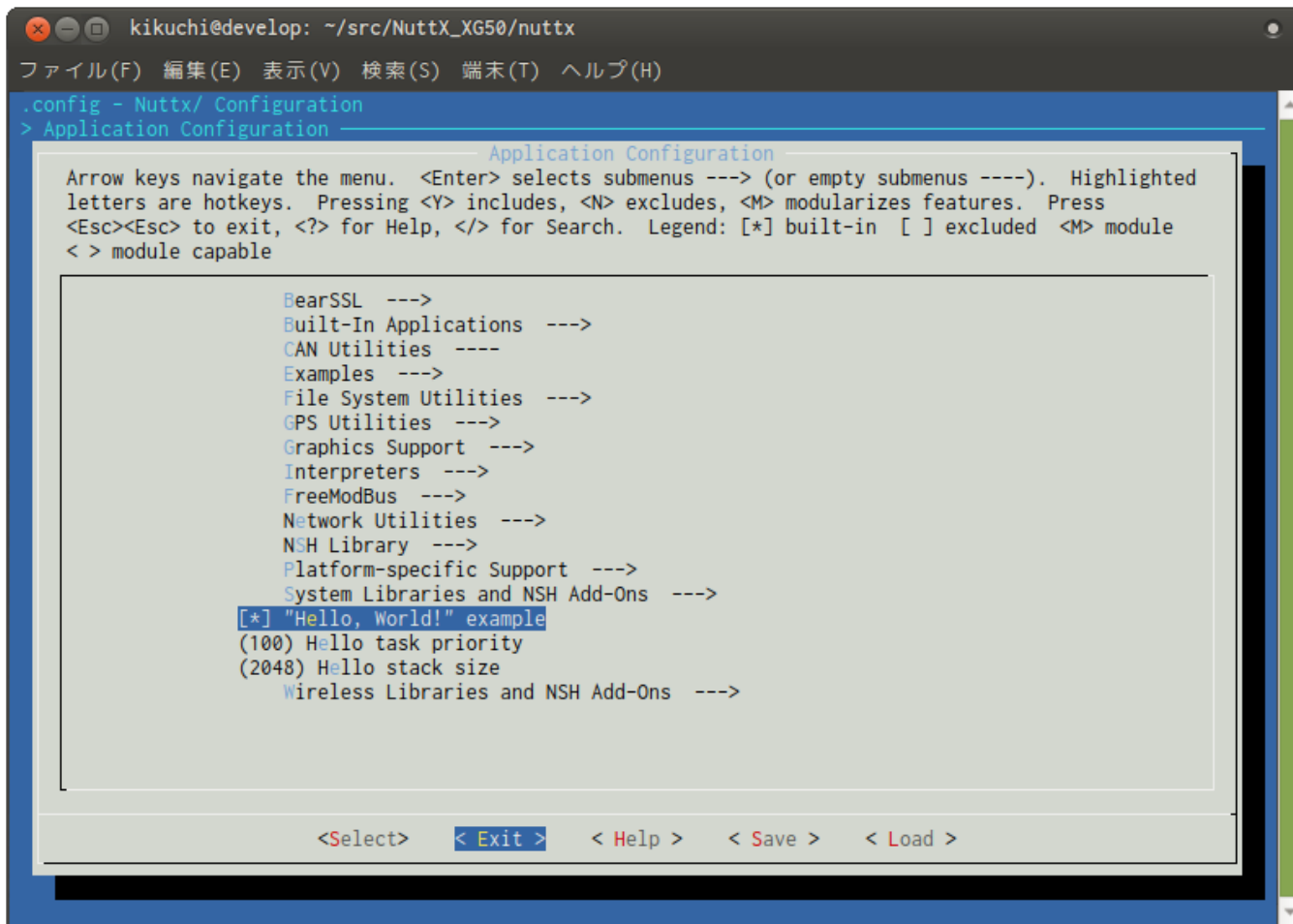


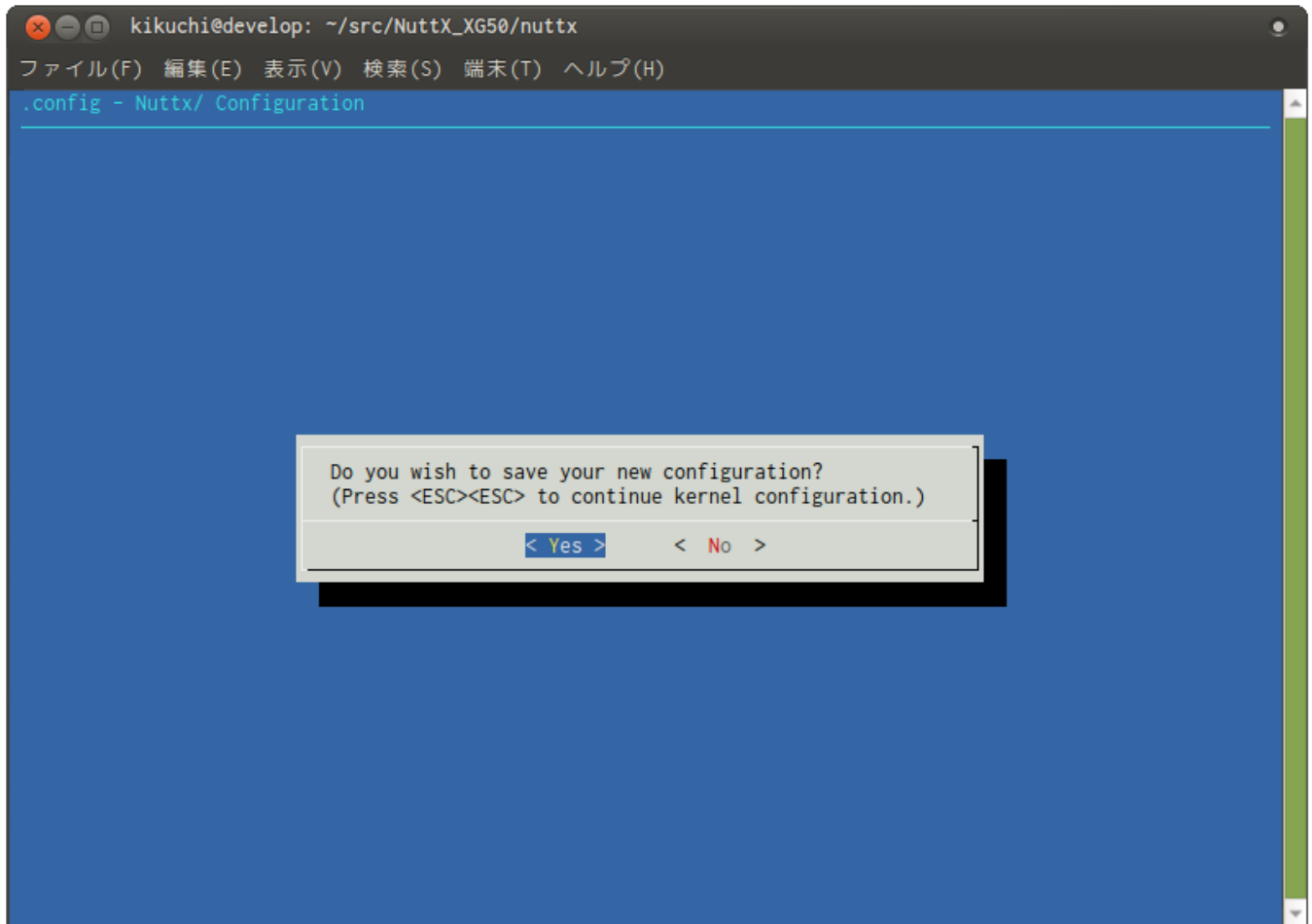
```
kikuchi@develop: ~/src/NuttX_XG50/nuttX
ファイル(F) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
.config - Nuttx/ Configuration
> Application Configuration
Application Configuration
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus ----). Highlighted
letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press
<Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module
< > module capable

BearSSL --->
Built-In Applications --->
CAN Utilities ----
Examples --->
File System Utilities --->
GPS Utilities --->
Graphics Support --->
Interpreters --->
FreeModBus --->
Network Utilities --->
NSH Library --->
Platform-specific Support --->
System Libraries and NSH Add-Ons --->
[*] "Hello, World!" example
(100) Hello task priority
(2048) Hello stack size
Wireless Libraries and NSH Add-Ons --->

<Select> <Exit> <Help> <Save> <Load>
```

<Exit> → **<Exit>** を選び、config の編集内容を反映させます。





ビルド

ビルドを行います。

```
develop:~/src/NuttX_XG50/nuttX$ make
... 略 ...
make[1]: Leaving directory '/home/kikuchi/src/NuttX_XG50/nuttX/arch/arm/src'
CP: nuttx.hex
CP: nuttx.bin
develop:~/src/NuttX_XG50/nuttX$
```

実機への書き込みと実行

[ファームウェアの書き込みと動作](#) と同じ手順で、XG-50 に書き込んで実行してみます。

```
NuttShell (NSH)
nsh> help
```

```
help usage: help [-v] [<cmd>]
```

[dirname	false	mkfatfs	pwd	time
?	date	free	mkfifo	reboot	true
basename	dd	help	mkrd	rm	uname
break	df	hexdump	mh	rmdir	umount
cat	dmesg	kill	mount	set	unset
cd	echo	ls	mv	sh	usleep
cp	exec	mb	mw	sleep	xd
cmp	exit	mkdir	ps	test	

Builtin Apps:

```
cu
hello <----- 増えている
i2c
sudo
nsh>
```

ビルトインアプリケーションとして **hello** が出てきました。
さっそく実行してみます。

```
nsh> hello
Hello, World!!
nsh>
```

きちんと動作しました！

1)

MessageQueue, Mutex, Semaphore などが利用できます。 [こちら](#) を参照してください。

2)

[こちら](#) を参照

From: <https://ma-tech.centurysys.jp/> - MA-X/MA-S/MA-E/IP-K Developers' Wiki

Permanent link: https://ma-tech.centurysys.jp/doku.php?id=xg_series_devel:add_builtin_command:start

Last update: 2020/11/30 09:55

