

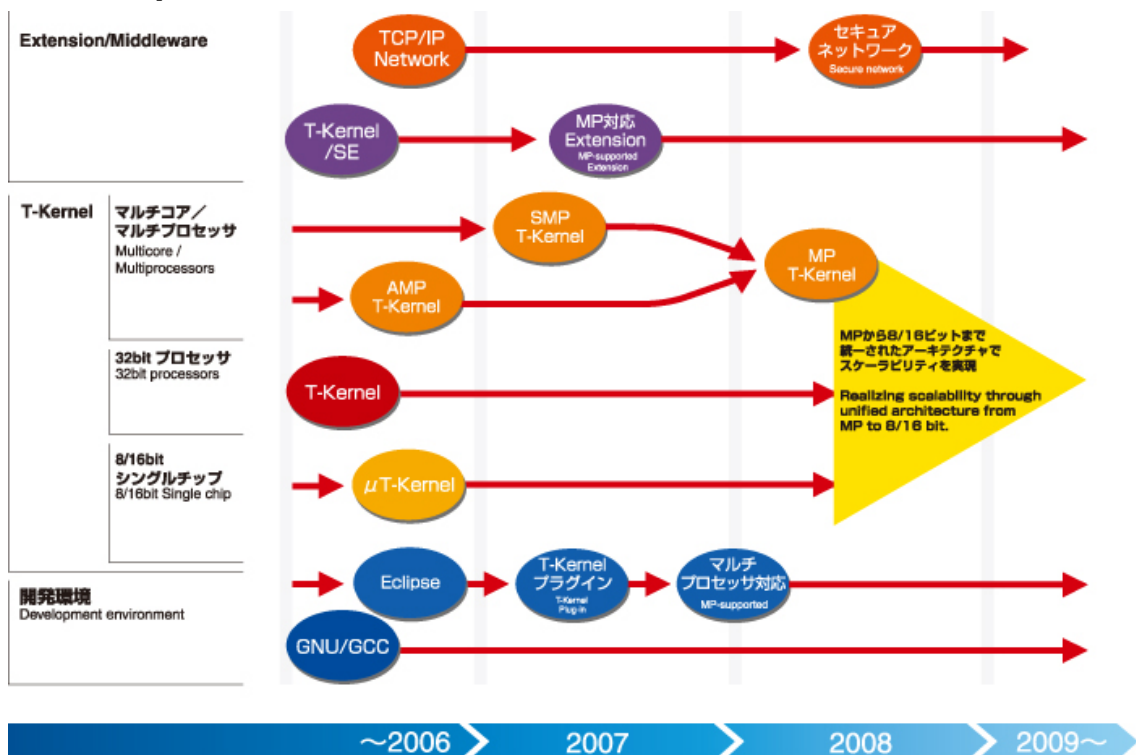
T-kernel의 디렉토리 구조 설명

저자 : 채승엽

mfcsource@msn.com

'06년 8월 9일 T-Engine 포럼에서 공개한 T-Kernel Version: 1.02.02을 기준으로 설명을 한다. 먼저, 간단하게 T-Engine포럼의 아래 [그림1]의 T-Kernel Loadmap을 설명하겠다.

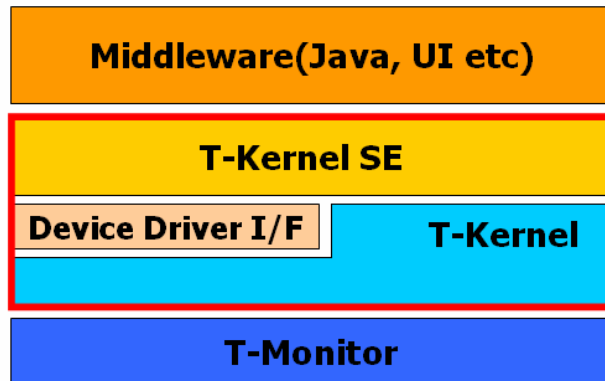
현재 공개된 T-Kernel은 아래의 빨간색의 T-Kernel이다. 그리고, T-Engine포럼 8/16bit Multiprocessors용 T-Kernel을 개발하여 일반인들에게 공개할 예정에 있다. 앞으로 복수의 cpu가 장착된 임베디드 관련 제품이 많아지게 될 전망이다.



[그림 1] T-Kernel Loadmap (<http://www.tron.org/topics/2007/2007-02-03.html>)

'07년 2월 현재 T-Engine포럼(<http://www.t-engine.org>)에서는 [그림 2]와 같은 빨간색 부분의 소스를 일반인에게 공개하였다. 아울러 '07년 3월 26일 uT-Kernel도 공개하였다.

T-Kernel	tkernel.1.02.02.tar.gz
T-Kernel SE	tkernel_se_1.00.00.tar.gz
Device Driver Interface	tkse_sample_drv_1.00.00.tar.gz



[그림 2] '07년 2월 현재 T-Engine포럼 오픈 소스 부분 (빨간색)

여기서, T-Monitor는 부트로더이며, Open Spec 형태로 현재는 문서만 공개하고 있다. Open Spec에 대해서 잠깐 설명을 하겠다. Open Spec은 소프트웨어 개발을 위한 사양서 문서이다. 예를 들면 난방이 잘 되는 집을 지을 때는 해가 동쪽에 뜨므로 창문으로 동쪽으로 한다. 그리고 난방이 잘 되기 위해 벽의 내부 구조는 어떻게 한다는 것과 같은 좋은 아이디어가 작성되어 있는 형태의 문서이다. 따라서, T-Engine 의 부트로더를 개발할 때 사양서를 기준으로, 기존의 리눅스 부트로더 등과 같이 T-Monitor 수정해서 개발하면 되겠다.

그리고 위의 [그림 2]를 보면 T-Kernel(t-kernel.1.02.02.tar.gz)소스에는 Device Driver Interface의 Library가 포함되어 있지 않다. Device Driver 개발시에는 Driver Device Interface의 Library(tkse_sample_drv_1.00.00.tar.gz)를 별도로 다운 받아서 구현하면 된다. 그리고 Device Driver 구동은 T-Kernel SE가 없어도 구동이 된다.

T-Engine포럼의 T-Kernel은 최소 용량의 Hard Real Time OS(μsec제어가 가능한 OS)이므로 커널이미지를 양산제품에 맞게 최소화 할 수 있다. T-Kernel 이외의 자세한 설명은 다음 칼럼에서 기술하도록 하겠다.

이번 칼럼에는 T-Kernel 부분에 대해서만 설명을 하겠다.

t-engine.org 홈페이지에서 t-kernel.1.02.02.tar.gz 다운로드 받아 압축을 풀면 다음과 같은 구성을 볼 수 있다.

tkernel_source		
	Config	설정 정보 (RomInfo, SYSCONF, DEVCONF)
	Etc	Makerule 등의 Perl 스크립트
	include	T-Kernel 관련 헤더 파일
	kernel	T-Kernel main source
	Lib	Library

그리고 tkernel_source에 포팅되어 있는 T-Engine포럼의 표준 하드웨어는 아래의 표와 같다.

Standard T-Engine/SH7727	SH3DSP
Standard T-Engine/SH7751R	SH4
Standard T-Engine/SH7760	SH4
Standard T-Engine/ARM720-S1C	ARM720T
Standard T-Engine/ARM920-MX1	ARM920T
Standard T-Engine/ARM926-MB8	ARM926J
Standard T-Engine/VR5500	MIPS4
micro T-Engine/VR4131	MIPS3
micro T-Engine/M32104	M32R



[그림 3] Standard T-Engine 보드(위) 및 micro T-Engine 보드(아래)

시판되고 있는 T-Engine 보드가 있으면 소스코드 수정없이 build하여 커널 이미지를 T-Engine보드에 다운로드하여 커널이미지가 실행시킬 수 있다. 그리고 일반적으로 국내에서 많이 사용되는 ARM관련 보드의 경우는 위 표의 ARM Core소스를 수정하여 포팅하면 되겠다. 현재 T-Engine 포럼 간사회원인 퍼스널미디어(주)에서는 x86 CPU에 T-Kernel 포팅하여 양산제품을 제작하였다. 그러므로 본 칼럼을 보시고 관심이 있는 사람은 기존 개발보드에 T-Kernel을 포팅하여 개발하면 되겠다.

이제 tkernel_source의 대표적인 디렉토리에 대하여 설명하도록 하겠다.

1. \$(BD)/config

config 디렉토리는 t-kernel에서 중요한 역할을 하는 부분이다. t-kernel을 build 하면 만들어진 커널이미지와 별개로 만들어지는 커널관련 설정정보 이미지 파일이다. 커널이미지 업로드와 별개로 설정정보 이미지를 업로드 하여 device driver 설정이나 커널설정 등을 커널이미지 수정없이 변경이 가능하다. 다른 임베디드 OS와 비교하여 설명하면 MS사의 Wince의 config.bib(register등록)와 비슷한 기능이며, 임베디드 리눅스 경우 임베디드 리눅스 부팅 후의 .bashprofile과 비슷한 기능이다. 즉, config 디렉토리에는 T-Kernel이 ROM(flash)에 탑재되어 있는 경우 ROM에서 RAM으로 커널이미지를 복사할 때의 정보를 포함하고 있다고 생각하면 된다.

rominfo.c에서 보면 아래와 같이 구조체가 선언되어 있다.

```
RomInfo rominfo = {
0, /* Reserved (not in use) */
RL_KERNEL_START, /* Kernel startup address */
RL_USERAREA_TOP, /* RAM user area top */
RL_RESETEINIT, /* Reset initialization program address */
RL_RDSK_TYPE, /* ROM disk type */
RL_RDSK_BLOCK, /* ROM disk block size*/
RL_RDSK_START, /* ROM disk start address */
RL_RDSK_END, /* ROM disk end address */
{0}, /* Reserved (always 0) */
RL_USERINIT, /* User initialization program address */
RL_SYSCONF, /* SYSCONF top */
RL_DEVCONF, /* DEVCONF top */
};
```

rominfo_conf.h

```
#ifndef _ROMINFO_CONF_
#define _ROMINFO_CONF_

IMPORT UB SYSCONF[], DEVCONF[];

/* Kernel address */
#define RL_KERNEL_START (FP)0x80020000 /* Kernel start address */
#define RL_SYSCONF (UB*)SYSCONF /* SYSCONF top */
```

```

#define RI_DEVCONF          (UB*)DEVCONF      /* DEVCONF top */

/* User definition */
#define RI_USERAREA_TOP    (VP)0x8e000000     /* RAM user area top */
#define RI_USERINIT        (FP)NULL          /* User initialization program */
#define RI_RESETINIT       (FP)NULL          /* Reset initialization program */

/* Rom disk */
#define RI_RDSK_TYPE       (UW)0            /* ROM disk type */
#define RI_RDSK_BLOCK      (UW)0            /* ROM disk block size */
#define RI_RDSK_START      (UW)0            /* ROM disk start address */
#define RI_RDSK_END        (UW)0            /* ROM disk end address */

#endif /* _ROMINFO_CONF_ */

```

tkernel_source/config/src/Makefile.common의 Perl scripts에 의해서 SYSCONF, DEVCONF파일을 sysconf.c, devconf.c파일로 구조체 형식으로 변환을 해준다.

SYSCONF System configuration file (SYSCONF)

```

#
#      Product information
#
TSysName      T-Kernel      # System name

#
#      Kernel version information for tk_ref_ver(T_RVER*)
#
Maker          0x0000      # = "T-Engine Forum"
ProductID      0x0000      # Kernel Identifier
SpecVer        0x7100      # = "T-Kernel" + "Ver 1.00"
ProductVer     0x0102      # Product Version
ProductNo      0x0000 0x0000 0x0000 0x0000
                # Product Number [0]-[3]

#
#      T-Kernel/OS
#

```

TMaxTskId	150	# Maximum task ID
TMaxSemId	100	# Maximum semaphore ID
TMaxFlgId	100	# Maximum event flag ID
TMaxMbxId	20	# Maximum mail box ID
TMaxMtxId	100	# Maximum mutex ID
TMaxMbfId	20	# Maximum message buffer ID
TMaxPorId	50	# Maximum rendezvous port ID
TMaxMpfId	10	# Maximum fixed size memory pool ID
TMaxMplId	10	# Maximum variable size memory pool ID
TMaxCycId	20	# Maximum cyclic handler ID
TMaxAlmId	40	# Maximum alarm handler ID
TMaxResId	60	# Maximum resource group ID
TMaxSsyId	50	# Maximum sub system ID
TMaxSsyPri	16	# Maximum sub system priority
TSysStkSz	2048	# Default system stack size (byte)
TSVCLimit	1	# SVC protection level
TTimPeriod	10	# Timer interval (msec)
#		
#	T-Kernel/SM	
#		
TMaxRegDev	32	# Maximum number of devices registration
TMaxOpnDev	64	# Maximum number of devices open
TMaxReqDev	64	# Maximum number of device requests
TDEvtMbfSz	1024 64	# Event notification message buffer size (byte), # Maximum length of message (byte)
#		
#	Task Event(1-8)	
#		
TEV_MsgEvt	1	# Message management : Recieve message
TEV_MsgBrk	2	# Message management : Release of an message waiting state
TEV_GDI	3	# GDI interface
TEV_FFLock	4	# Release of an FIFO lock waiting state
#		

```
#      Segment manager
#
RealMemEnd    0x90000000    # RAM bottom address (logical address)
```

SYSCONF는 T-Kernel 자원의 제한적인 설정 및 미들웨어(t-shell 등) 설정의 역할을 한다.

DEVCONF Device configuration file (DEVCONF)

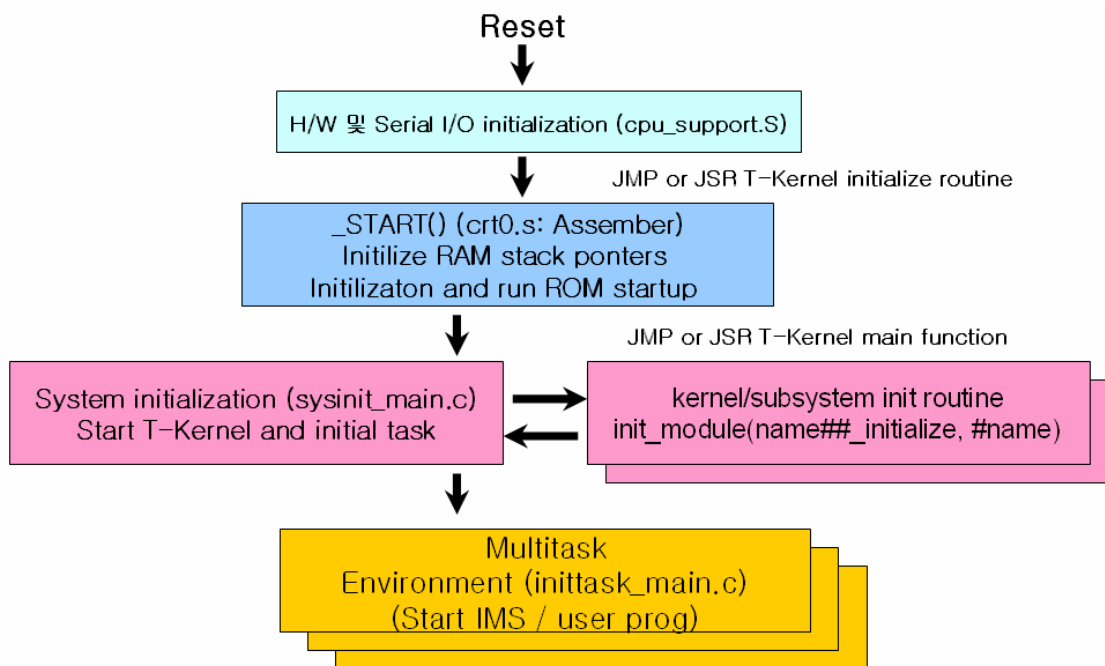
```
#      Device configuration
#
#
#
# Use DCTAG_XXX as a parameter of GetDevConf().
# DCTAG_XXX is defined in <sys/rominfo.h>.
#
#
# Debug mode (1:debug mode, 0:normal mode)
DEBUGMODE    1
```

DEVCONF는 Device Driver에 대한 설정을 하며, 실시간으로 Device Driver 설정을 변경할 수 있는 곳이다.

config디렉토리에서 build하면 rominfo.mot가 만들어지고 이것을 ROM에 저장한다. 그렇다면 rominfo.mot와 T-Monitor/T-Kernel과는 어떠한 관계가 있는지 설명을 하겠다.

Rominfo.mot는 T-Kernel source의 특징적인 부분이라고 말할 수 있다.

아래 [그림4]와 같이 개발자가 직접 작성해야 하는 부분이다. 아래를 확인해 보면 모든 디렉토리 경로에 {target_system}가 포함 되어 있다. target_system은 개발보드에서 사용되는 CPU를 말한다. 새로운 개발보드에 T-Kernel을 포팅을 하려면 이부분을 반드시 수정해야 한다.



[그림 6] T-Kernel Booting 순서

2. \$(BD)/etc



[그림 7] ‘\$(BD)/etc/’의 파일 구조

T-Kernel 소스 및 새로운 기능을 build 하기 위한 Perl Scripts이다. 그러므로 반드시 문서 편집기를 이용하여 상세히 분석해야 할 부분이다. 개발과정에서 이 부분을 수정하는 일은 없다. 그러나 이 Perl Scripts를 잘 사용해야지 새로운 device driver 및 미들웨어를 손쉽게 개발할 수 있다. 사용 방법에 대해서는 다음 칼럼에서 설명하겠다. 리눅스 셸커맨드에서 사용하는 명령어라고 생각하면 되겠다. 모든 임베디드 OS 관련 부분을 개발하기 위해서는 별도로 반드시 Perl 프로그램에 대해서 공부를 하기 바란다.

etc디렉토리	Perl Scripts 기능 설명
backup_copy	T-Kernel build 과정에서 생성된 파일을 복사함
makerules	T-Kernel 개발환경의 build 경로 지정

mkiflib	T-Kernel에서 Extended SVC Handler 만들 때 사용. 매크로가 작성된 1개의 *.h의 참조하여 *.S의 어셈블리어를 생성.
mktcsvc	T-Kernel의 API인 td_XXX_XXX의 *.S 파일을 만듬.
mktksvc	T-Kernel의 API인 tk_XXX_XXX의 *.S 파일을 만듬.
platform	Host 개발환경 즉, PC 환경에서 사용 OS 지정

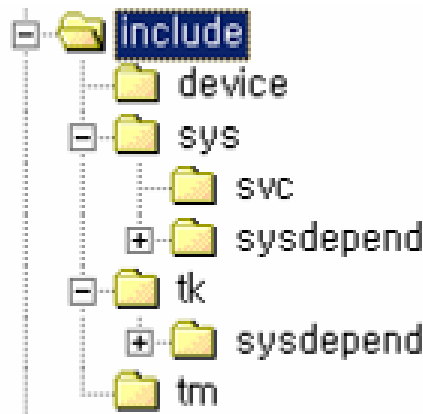
'\$(BD)/etc'에는 *.S, fn*.h, if*.h을 만들게 해주는 perl Scripts이 들어 있다.

mkiflib : Extended SVC
mktksvc : T-Kernel/OS SVC
mktcsvc : T-Kernel/SM SVC

그리고 위와 비슷한 기능으로 '\$(BD)/lib/libsvc/build/sysdepend/{target_system}'에는 Assamble을 생성할 때 참조하는 perl Scripts이 들어 있다.

makeifex.pl : Extended SVC
makeiftk.pl : T-Kernel/OS SVC
makeiftd.pl : T-Kernel/SM SVC

3. \$(BD)/include

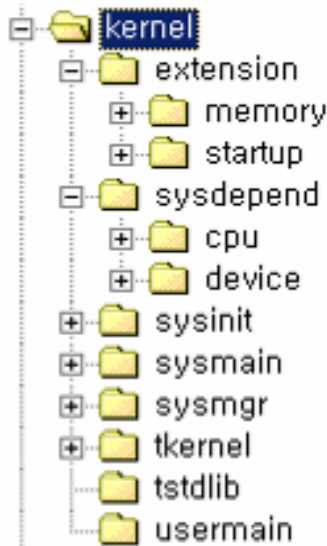


[그림 8] \$(BD)/include 디렉토리 구조

include디렉토리는 T-Kernel나 user 프로그램에서 사용되는 각종 헤더 파일이 있다. 임베디드 리눅스의 경우는 include디렉토리에 수많은 하위 디렉토리 구조로 되어 있기 때문에 분석하는데 많은 시간이 걸리지만 T-Kernel의 include 파일은 그다지 많지 않으므로 시간이 있으면 해당 API가 정의된 함수구조를 분석해 보기 바란다. 황색으로 된 부분이 제품 개발시에 제일 중요한 부분이다.

include 디렉토리	기능 설명
device	디바이스 드라이버 관련의 헤더파일 정의하는 부분. 공개된 T-Kernel 소스에서는 'devconf.h'밖에 없지만 Device Driver 개발할 때 관련 헤더파일은 여기에 헤더 파일을 작성해야 함. screen.h, usb.h 등의 헤더파일을 작성.
sys	T-Kernel 시스템 내부의 정보에 관한 헤더 파일을 있음.
svc	Extended SVC 관련의 헤더파일임. Subsystem이라는 T-Kernel 특유의 기능이며 Device Driver 및 미들웨어 개발할 때 헤더파일을 정의함. fn*.h, if*.h의 헤더 파일이 정의 되어 있는데, 이것은 mkiflib로 만듦.
sysdepend	시스템 정보의 하드웨어 의존부의 헤더 파일임.
tk	T-Kernel 관련의 헤더 파일을 정의함.
sysdepend	하드웨어 의존부의 헤더 파일임.
tm	T-Monitor 및 디버그 관련의 헤더 파일이며, T-Monitor개발 할 때 사용하는 헤더파일.

4. \$(BD)/kernel



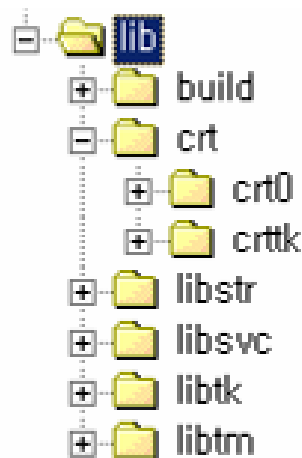
[그림 9] \$(BD)/kernel 디렉토리 구조

kernel 디렉토리의 커널 기본부는 sysinit, sysmain, sysmgr, tkernel, tstdlib의 5개 디렉토리며 아래의 황색 부분이다. 이 부분은 T-License에서 T-Engine포럼에서만 수정 배포할 수 있는 부분이다. kernel 디렉토리내의 하드웨어 의존부는 extension, sysdepend의 2개의 디렉토리며 아래의 보라색 부분이다.

실제 제품 개발 시에 제일 중요한 부분은 아래의 파란색 부분인 ‘usermain’ 디렉토리다.

kernel 디렉토리	기능 설명
extension	T-Kernel SE 등 상위의 소프트웨어에 관련하는 하드웨어 의존부.
memory	메모리 관리 부분. MMU를 사용하지 않고 실제 메모리만으로 동작하는 시스템에 대해서는 ‘extension/memory/nommu’ 디렉토리 작업하면 됨.
	startup
sysdepend	하드웨어에 의존하는 소스 코드
cpu	타겟 시스템의 CPU에 의존하는 소스 코드
	device
sysinit	T-Kernel 의 초기화/종료 처리 루틴.
sysmain	T-Kernel 전체의 구축 디렉토리.
sysmgr	T-Kernel/SM 소스 코드. T-Kernel 시스템 관리 부분
tkernel	T-Kernel/OS 소스 코드. T-Kernel의 커널 API 부분
tstdlib	T-Kernel내에서 사용하고 있는 비트 조작 함수의 소스 코드입니다
usermain	user 프로그램을 실행 시키는 부분으로 T-Kernel 동작이 되며 이 부분이 가장 나중에 실행 된다. 새롭게 작성한 Device Driver 로딩이나 원하는 GUI 프로그램의 실행 함수를 로딩 시켜 주면 됨.

5. \$(BD)/lib



[그림 10] \$(BD)/lib 디렉토리 구조

lib 디렉토리는 T-Kernel나 user 프로그램 개발에 필요한 library를 제공하는 부분이다. 그리고, Device Driver interface나 미들웨어 개발에 필요한 library를 별도로 작성하여 추가하면 된다.

Device Driver interface library는 “tkse_sample_drv_1.00.00.tar.gz”로 압축을 풀면 “\$(BD)/lib/libdrvif” 디렉토리가 제공된다.

lib 디렉토리에 제공되는 library 함수에 대해서는 전체를 분석하여 제품을 개발 시에 활용하면 된다. 한가지 예로 Device Driver나 미들웨어를 개발할 때 필요한 함수를 직접 개발할 때가 있다. 그때 library 함수에서 제공하는 함수를 사용할 수 있겠다.

개발자가 제품을 개발할 때 library 함수를 잘 분석하지 못하여 별도로 필요한 함수를 만드는 경우가 많이 발생한다. 그렇게 되면 전체 커널이미지가 커지게 된다. 그렇게 되면 Hard Real Time OS인 T-Kernel에서는 Hard Real Time을 보장하지 못하거나 불필요한 메모리 자원의 낭비가 발생할 수도 있다.

lib 디렉토리	기능 설명
build	전체 lib디렉토리에서 target_system 필요한 library를 make를 실행하여 만듦.
crt	T-Kernel 관련 startup 루틴 library
	crt0 공통의 startup 루틴. 모든 user 프로그램의 startup 루틴으로서 사용됨.
	crttk T-Kernel의 기능을 이용하는 프로그램의 startup 루틴. 주로 Device Driver나 시스템 관리 등의 시스템 프로그램의 startup 루틴으로 사용됨.
libstr	T-Kernel 및 library에서 사용하고 있는 표준적인 라이브러리 함수로 주로 변수 라인 조작 관계의 소스 코드.
libsvc	T-Kernel의 시스템 콜의 interface library.
libtk	T-Kernel가 제공하는 함수 library.
libtm	T-Monitor의 시스템 콜의 interface library로 T-Monitor개발에 사용.

T-Kernel Build하기

리눅스나 Cygwin에서 공개된 T-Kernel을 아래의 순서로 build하면 kernel-rom.mot 라는 커널 이미지파일이 생성된다. (.mot : Motrola S-recode)

물론 build의 makefile을 수정하여 임베디드 리눅스처럼 kernel-rom.bin 과 같은 바이너리 포맷으로 만들 수도 있다.

kernel-rom.mot 만들기

```
% cd ~/tkbuild
% tar xvfz tkernel.x.xx.xx.tar.gz (x.xx.xx 는 버전 번호)
```

```
% export BD=~/tkbuild/tkernel_source (bash의 경우)

% cd ~/tkbuild/tkernel_source/lib/build/std_sh7727
% make

% cd ~/tkbuild/tkernel_source/kernel/sysmain/build/std_sh7727
% make
```

rominfo.mot 만들기

```
% cd ~/tkbuild/tkernel_source/config/build/std_sh7727
% make
```