# 1. リファレンス

## 1.1. サービスコール一覧

(1) タスク管理機能

ER_ID tskid = acre_tsk(const T_CTSK *pk_ctsk) 〔TD〕
ER ercd = sac_tsk(ID tskid, const ACVCT *p_acvct) 〔TPD〕
ER ercd = del_tsk(ID tskid) 〔TD〕
ER ercd = act_tsk(ID tskid) 〔T〕
ER ercd = iact_tsk(ID tskid) 〔I〕
ER ercd = mact_tsk(ID tskid, ID prcid) 〔TM〕
ER ercd = imact_tsk(ID tskid, ID prcid) 〔IM〕
ER_UINTactcnt = can_act(ID tskid) 〔T〕
ER ercd = mig_tsk(ID tskid, ID prcid) 〔TM〕
ER ercd = ext_tsk() 〔T〕
ER ercd = ter_tsk(ID tskid) 〔T〕
ER ercd = chg_pri(ID tskid, PRI tskpri) 〔T〕
ER ercd = get_pri(ID tskid, PRI *p_tskpri) 〔T〕
ER ercd = get_inf(intptr_t *p_exinf) 〔T〕
ER ercd = ref_tsk(ID tskid, T_RTSK *pk_rtsk) 〔T〕

(2) タスク付属同期機能

ER ercd = slp_tsk() 〔T〕
ER ercd = tslp_tsk(TMO tmout) 〔T〕
ER ercd = wup_tsk(ID tskid) 〔T〕
ER ercd = iwup_tsk(ID tskid) 〔I〕
ER_UINT wupcnt = can_wup(ID tskid) 〔T〕
ER ercd = rel_wai(ID tskid) 〔T〕
ER ercd = irel_wai(ID tskid) 〔I〕
ER ercd = sus_tsk(ID tskid) 〔T〕
ER ercd = rsm_tsk(ID tskid) 〔T〕
ER ercd = dis_wai(ID tskid) 〔TP〕
ER ercd = idis_wai(ID tskid) 〔IP〕
ER ercd = ena_wai(ID tskid) 〔TP〕
ER ercd = iena_wai(ID tskid) 〔IP〕
ER ercd = dly_tsk(RELTIM dlytim) 〔T〕

(3) タスク例外処理機能

ER ercd = def_tex(ID tskid, const T_DTEX *pk_dtex) 〔TD〕
ER ercd = ras_tex(ID tskid, TEXPTN rasptn) 〔T〕
ER ercd = iras_tex(ID tskid, TEXPTN rasptn) 〔I〕

ER ercd = dis_tex() 〔T〕
ER ercd = ena_tex() 〔T〕
bool_t state = sns_tex() 〔TI〕
ER ercd = ref_tex(ID tskid, T_RTEX *pk_rtex) 〔T〕

(4) 同期・通信機能

セマフォ

ER_ID semid = acre_sem(const T_CSEM *pk_csem) 〔TD〕
ER ercd = sac_sem(ID semid, const ACVCT *p_acvct) 〔TPD〕
ER ercd = del_sem(ID semid) 〔TD〕
ER ercd = sig_sem(ID semid) 〔T〕
ER ercd = isig_sem(ID semid) 〔I〕
ER ercd = wai_sem(ID semid) 〔T〕
ER ercd = pol_sem(ID semid) 〔T〕
ER ercd = twai_sem(ID semid, TMO tmout) 〔T〕
ER ercd = ini_sem(ID semid) 〔T〕
ER ercd = ref_sem(ID semid, T_RSEM *pk_rsem) 〔T〕

イベントフラグ

ER_ID flgid = acre_flg(const T_CFLG *pk_cflg) 〔TD〕
ER ercd = sac_flg(ID flgid, const ACVCT *p_acvct) 〔TPD〕
ER ercd = del_flg(ID flgid) 〔TD〕
ER ercd = set_flg(ID flgid, FLGPTN setptn) 〔T〕
ER ercd = iset_flg(ID flgid, FLGPTN setptn) 〔I〕
ER ercd = clr_flg(ID flgid, FLGPTN clrptn) 〔T〕
ER ercd = wai_flg(ID flgid, FLGPTN waiptn, MODE wfmode, FLGPTN *p_flgptn) 〔T〕
ER ercd = pol_flg(ID flgid, FLGPTN waiptn, MODE wfmode, FLGPTN *p_flgptn) 〔T〕
ER ercd = twai_flg(ID flgid, FLGPTN waiptn, MODE wfmode, FLGPTN *p_flgptn, TMO tmout) 〔T〕
ER ercd = ini_flg(ID flgid) 〔T〕
ER ercd = ref_flg(ID flgid, T_RFLG *pk_rflg) 〔T〕

データキュー

ER_ID dtqid = acre_dtq(const T_CDTQ *pk_cdtq) 〔TD〕
ER ercd = sac_dtq(ID dtqid, const ACVCT *p_acvct) 〔TPD〕
ER ercd = del_dtq(ID dtqid) 〔TD〕
ER ercd = snd_dtq(ID dtqid, intptr_t data) 〔T〕
ER ercd = psnd_dtq(ID dtqid, intptr_t data) 〔T〕
ER ercd = ipsnd_dtq(ID dtqid, intptr_t data) 〔I〕
ER ercd = tsnd_dtq(ID dtqid, intptr_t data, TMO tmout) 〔T〕
ER ercd = fsnd_dtq(ID dtqid, intptr_t data) 〔T〕
ER ercd = ifsnd_dtq(ID dtqid, intptr_t data) 〔I〕
ER ercd = rcv_dtq(ID dtqid, intptr_t *p_data) 〔T〕

ER ercd = prcv_dtq(ID dtqid, intptr_t *p_data) 〔T〕
ER ercd = trcv_dtq(ID dtqid, intptr_t *p_data, TMO tmout) 〔T〕
ER ercd = ini_dtq(ID dtqid) 〔T〕
ER ercd = ref_dtq(ID dtqid, T_RDTQ *pk_rdtq) 〔T〕

優先度データキュー

ER_ID pdqid = acre_pdq(const T_CPDQ *pk_cpdq) 〔TD〕
ER ercd = sac_pdq(ID pdqid, const ACVCT *p_acvct) 〔TPD〕
ER ercd = del_pdq(ID pdqid) 〔TD〕
ER ercd = snd_pdq(ID pdqid, intptr_t data, PRI datapri) 〔T〕
ER ercd = psnd_pdq(ID pdqid, intptr_t data, PRI datapri) 〔T〕
ER ercd = ipsnd_pdq(ID pdqid, intptr_t data, PRI datapri) 〔I〕
ER ercd = tsnd_pdq(ID pdqid, intptr_t data, PRI datapri, TMO tmout) 〔T〕
ER ercd = rcv_pdq(ID pdqid, intptr_t *p_data, PRI *p_datapri) 〔T〕
ER ercd = prcv_pdq(ID pdqid, intptr_t *p_data, PRI *p_datapri) 〔T〕
ER ercd = trcv_pdq(ID pdqid, intptr_t *p_data,PRI *p_datapri, TMO tmout) 〔T〕
ER ercd = ini_pdq(ID pdqid) 〔T〕
ER ercd = ref_pdq(ID pdqid, T_RPDQ *pk_rpdq) 〔T〕

メールボックス

ER_ID mbxid = acre_mbx(const T_CMBX *pk_cmbx) 〔TDp〕
ER ercd = del_mbx(ID mbxid) 〔TDp〕
ER ercd = snd_mbx(ID mbxid, T_MSG *pk_msg) 〔Tp〕
ER ercd = rcv_mbx(ID mbxid, T_MSG **ppk_msg) 〔Tp〕
ER ercd = prcv_mbx(ID mbxid, T_MSG **ppk_msg) 〔Tp〕
ER ercd = trcv_mbx(ID mbxid, T_MSG **ppk_msg, TMO tmout) 〔Tp〕
ER ercd = ini_mbx(ID mbxid) 〔Tp〕
ER ercd = ref_mbx(ID mbxid, T_RMBX *pk_rmbx) 〔Tp〕

ミューテックス

ER_ID mtxid = acre_mtx(const T_CMTX *pk_cmtx) 〔TD〕
ER ercd = sac_mtx(ID mtxid, const ACVCT *p_acvct) 〔TPD〕
ER ercd = del_mtx(ID mtxid) 〔TD〕
ER ercd = loc_mtx(ID mtxid) 〔T〕
ER ercd = ploc_mtx(ID mtxid) 〔T〕
ER ercd = tloc_mtx(ID mtxid, TMO tmout) 〔T〕
ER ercd = unl_mtx(ID mtxid) 〔T〕
ER ercd = ini_mtx(ID mtxid) 〔T〕
ER ercd = ref_mtx(ID mtxid, T_RMTX *pk_rmtx) 〔T〕

メッセージバッファ

ER_ID mbfid = acre_mbf(const T_CMBF *pk_cmbf) 〔TD〕
ER ercd = sac_mbf(ID mbfid, const ACVCT *p_acvct) 〔TPD〕

ER ercd = del_mbf(ID mbfid) 〔TD〕
ER ercd = snd_mbf(ID mbfid, const void *msg, uint_t msgsz) 〔T〕
ER ercd = psnd_mbf(ID mbfid, const void *msg, uint_t msgsz) 〔T〕
ER ercd = tsnd_mbf(ID mbfid, const void *msg, uint_t msgsz, TMO tmout) 〔T〕
ER_UINT msgsz = rcv_mbf(ID mbfid, void *msg) 〔T〕
ER_UINT msgsz = prcv_mbf(ID mbfid, void *msg) 〔T〕
ER_UINT msgsz = trcv_mbf(ID mbfid, void *msg, TMO tmout) 〔T〕
ER ercd = ini_mbf(ID mbfid) 〔T〕
ER ercd = ref_mbf(ID mbfid, T_RMBF *pk_rmbf) 〔T〕

スピンロック

ER_ID spnid = acre_spn(const T_CSPN *pk_cspn) 〔TMD〕
ER ercd = sac_spn(ID spnid, const ACVCT *p_acvct) 〔TPMD〕
ER ercd = del_spn(ID spnid) 〔TMD〕
ER ercd = loc_spn(ID spnid) 〔TM〕
ER ercd = iloc_spn(ID spnid) 〔IM〕
ER ercd = try_spn(ID spnid) 〔TM〕
ER ercd = itry_spn(ID spnid) 〔IM〕
ER ercd = unl_spn(ID spnid) 〔TM〕
ER ercd = iunl_spn(ID spnid) 〔IM〕
ER ercd = ref_spn(ID spnid, T_RSPN *pk_rspn) 〔TM〕

(5) メモリプール管理機能

固定長メモリプール

ER_ID mpfid = acre_mpf(const T_CMPF *pk_cmpf) 〔TD〕
ER ercd = sac_mpf(ID mpfid, const ACVCT *p_acvct) 〔TPD〕
ER ercd = del_mpf(ID mpfid) 〔TD〕
ER ercd = get_mpf(ID mpfid, void **p_blk)** 〔**T**〕
**ER ercd** = **pget_mpf(ID mpfid, void** p_blk) 〔T〕
ER ercd = tget_mpf(ID mpfid, void **p_blk, TMO tmout) 〔T〕
ER ercd = rel_mpf(ID mpfid, void *blk) 〔T〕
ER ercd = ini_mpf(ID mpfid) 〔T〕
ER ercd = ref_mpf(ID mpfid, T_RMPF *pk_rmpf) 〔T〕

(6) 時間管理機能

システム時刻管理

ER ercd = get_tim(SYSTIM *p_systim) 〔T〕
ER ercd = get_utm(SYSUTM *p_sysutm) 〔TI〕

周期ハンドラ

ER_ID cycid = acre_cyc(const T_CCYC *pk_ccyc) 〔TD〕
ER ercd = sac_cyc(ID cycid, const ACVCT *p_acvct) 〔TPD〕

ER ercd = del_cyc(ID cycid) 〔TD〕
ER ercd = sta_cyc(ID cycid) 〔T〕
ER ercd = msta_cyc(ID cycid, ID prcid) 〔TM〕
ER ercd = stp_cyc(ID cycid) 〔T〕
ER ercd = ref_cyc(ID cycid, T_RCYC *pk_rcyc) 〔T〕

アラームハンドラ

ER_ID almid = acre_alm(const T_CALM *pk_calm) 〔TD〕
ER ercd = sac_alm(ID almid, const ACVCT *p_acvct) 〔TPD〕
ER ercd = del_alm(ID almid) 〔TD〕
ER ercd = sta_alm(ID almid, RELTIM almtim) 〔T〕
ER ercd = ista_alm(ID almid, RELTIM almtim) 〔I〕
ER ercd = msta_alm(ID almid, RELTIM almtim, ID prcid) 〔TM〕
ER ercd = imsta_alm(ID almid, RELTIM almtim, ID prcid) 〔IM〕
ER ercd = stp_alm(ID almid) 〔T〕
ER ercd = istp_alm(ID almid) 〔I〕
ER ercd = ref_alm(ID almid, T_RALM *pk_ralm) 〔T〕

オーバランハンドラ

ER ercd = def_ovr,(const T_DOVR *pk_dovr) 〔TD〕
ER ercd = sta_ovr(ID tskid, OVRTIM ovrtim) 〔T〕
ER ercd = ista_ovr(ID tskid, OVRTIM ovrtim) 〔I〕
ER ercd = stp_ovr(ID tskid) 〔T〕
ER ercd = istp_ovr(ID tskid) 〔I〕
ER ercd = ref_ovr(ID tskid, T_ROVR *pk_rovr) 〔T〕

(7) システム状態管理機能

ER ercd = sac_sys(const ACVCT *p_acvct) 〔TPD〕
ER ercd = rot_rdq(PRI tskpri) 〔T〕
ER ercd = irot_rdq(PRI tskpri) 〔I〕
ER ercd = mrot_rdq(PRI tskpri, ID prcid) 〔TM〕
ER ercd = imrot_rdq(PRI tskpri, ID prcid) 〔IM〕
ER ercd = get_tid(ID *p_tskid) 〔T〕
ER ercd = iget_tid(ID *p_tskid) 〔I〕
ER ercd = get_did(ID *p_domid) 〔TP〕
ER ercd = get_pid(ID *p_prcid) 〔TM〕
ER ercd = iget_pid(ID *p_prcid) 〔IM〕
ER ercd = loc_cpu() 〔T〕
ER ercd = iloc_cpu() 〔I〕
ER ercd = unl_cpu() 〔T〕
ER ercd = iunl_cpu() 〔I〕
ER ercd = dis_dsp() 〔T〕
ER ercd = ena_dsp() 〔T〕

bool_t state = sns_ctx() 〔TI〕
bool_t state = sns_loc() 〔TI〕
bool_t state = sns_dsp() 〔TI〕
bool_t state = sns_dpn() 〔TI〕
bool_t state = sns_ker() 〔TI〕
ER ercd = ext_ker() 〔TI〕
ER ercd = ref_sys(T_RSYS *pk_rsys) 〔T〕

(8) メモリオブジェクト管理機能

ER ercd = att_mem(const T_AMEM *pk_amem) 〔TPD〕
ER ercd = att_pma(const T_AMEM *pk_apma) 〔TPD〕
ER ercd = sac_mem(const void *base, const ACVCT *p_acvct) 〔TPD〕
ER ercd = det_mem(const void *base) 〔TPD〕
ER ercd = prb_mem(const void *base, SIZE size,ID tskid, MODE pmmode) 〔TP〕
ER ercd = ref_mem(const void *base, T_RMEM *pk_rmem) 〔TP〕

(9) 割込み管理機能

ER ercd = cfg_int(INTNO intno, const T_CINT *pk_cint) 〔TD〕
ER_ID isrid = acre_isr(const T_CISR *pk_cisr) 〔TD〕
ER ercd = sac_isr(ID isrid, const ACVCT *p_acvct) 〔TPD〕
ER ercd = del_isr(ID isrid) 〔TD〕
ER ercd = ref_isr(ID isrid, T_RISR *pk_risr) 〔T〕
ER ercd = def_inh(INHNO inhno, const T_DINH *pk_dinh) 〔TD〕
ER ercd = dis_int(INTNO intno) 〔T〕
ER ercd = ena_int(INTNO intno) 〔T〕
ER ercd = ref_int(INTNO intno, T_RINT *pk_rint) 〔T〕
ER ercd = chg_ipm(PRI intpri) 〔T〕
ER ercd = get_ipm(PRI *p_intpri) 〔T〕

(10) CPU例外管理機能

ER ercd = def_exc(EXCNO excno, const T_DEXC *pk_dexc) 〔TD〕
bool_t stat = xsns_dpn(void *p_excinf) 〔TI〕
bool_t stat = xsns_xpn(void *p_excinf) 〔TI〕

(11) 拡張サービスコール管理機能

ER ercd = def_svc(FN fncd, const T_DSVC *pk_dsvc) 〔TPD〕
ER_UINT ercd = cal_svc(FN fncd, intptr_t par1, intptr_t par2,intptr_t par3, intptr_t par4, intptr_t par5) 〔TIP〕

(12) システム構成管理機能

ER ercd = ref_cfg(T_RCFG *pk_rcfg) 〔T〕
ER ercd = ref_ver(T_RVER *pk_rver) 〔T〕

## 1.2. 静的API一覧

(1) タスク管理機能

＊保護機能対応でないカーネルの場合

CRE_TSK(ID tskid, { ATR tskatr, intptr_t exinf, TASK task, PRI itskpri, SIZE stksz, STK_T *stk }) 〔S〕

＊保護機能対応カーネルの場合

CRE_TSK(ID tskid, { ATR tskatr, intptr_t exinf, TASK task, PRI itskpri, SIZE stksz, STK_T *stk,SIZE sstksz, STK_T *sstk }) 〔SP〕 sstksz およびsstkの記述は省略することができる.

AID_TSK(uint_t notsk) 〔SD〕
SAC_TSK(ID tskid, { ACPTN acptn1, ACPTN acptn2,ACPTN acptn3, ACPTN acptn4 }) 〔SP〕
DEF_EPR(ID tskid, { PRI exepri }) 〔S〕

(2) タスク付属同期機能

なし

(3) タスク例外処理機能

DEF_TEX(ID tskid, { ATR texatr, TEXRTN texrtn }) 〔S〕

(4) 同期・通信機能

セマフォ

CRE_SEM(ID semid, { ATR sematr, uint_t isemcnt, uint_t maxsem }) 〔S〕
AID_SEM(uint_t nosem) 〔SD〕
SAC_SEM(ID semid, { ACPTN acptn1, ACPTN acptn2,ACPTN acptn3, ACPTN acptn4 }) 〔SP〕

イベントフラグ

CRE_FLG(ID flgid, { ATR flgatr, FLGPTN iflgptn }) 〔S〕
AID_FLG(uint_t noflg) 〔SD〕
SAC_FLG(ID flgid, { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 }) 〔SP〕

データキュー

CRE_DTQ(ID dtqid, { ATR dtqatr, uint_t dtqcnt, void *dtqmb }) 〔S〕
AID_DTQ(uint_t nodtq) 〔SD〕
SAC_DTQ(ID dtqid, { ACPTN acptn1, ACPTN acptn2,ACPTN acptn3, ACPTN acptn4 }) 〔SP〕

優先度データキュー

CRE_PDQ(ID pdqid, { ATR pdqatr, uint_t pdqcnt, PRI maxdpri, void *pdqmb }) 〔S〕
AID_PDQ(uint_t nopdq) 〔SD〕
SAC_PDQ(ID pdqid, { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 }) 〔SP〕

メールボックス

CRE_MBX(ID mbxid, { ATR mbxatr, PRI maxmpri, void *mprihd }) 〔Sp〕
AID_MBX(uint_t nombx) 〔SpD〕

ミューテックス

CRE_MTX(ID mtxid, { ATR mtxatr, PRI ceilpri }) 〔S〕
AID_MTX(uint_t nomtx) 〔SD〕
SAC_MTX(ID mtxid, { ACPTN acptn1, ACPTN acptn2,ACPTN acptn3, ACPTN acptn4 }) 〔SP〕

メッセージバッファ

CRE_MBF(ID mbfid, { ATR mbfatr, uint_t maxmsz, SIZE mbfsz, void *mbfmb }) 〔S〕
AID_MBF(uint_t nombf) 〔SD〕
SAC_MBF(ID mbfid, { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 }) 〔SP〕

スピンロック

CRE_SPN(ID spnid, { ATR spnatr }) 〔SM〕
AID_SPN(uint_t nospn) 〔SMD〕
SAC_SPN(ID spnid, { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 }) 〔SPM〕

(5) メモリプール管理機能

固定長メモリプール

CRE_MPF(ID mpfid, { ATR mpfatr, uint_t blkcnt, uint_t blksz, MPF_T *mpf, void *mpfmb }) 〔S〕
AID_MPF(uint_t nompf) 〔SD〕
SAC_MPF(ID mpfid, { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 }) 〔SP〕

(6) 時間管理機能

周期ハンドラ

CRE_CYC(ID cycid, { ATR cycatr, intptr_t exinf, CYCHDR cychdr, RELTIM cyctim, RELTIM cycphs }) 〔S〕
AID_CYC(uint_t nocyc) 〔SD〕
SAC_CYC(ID cycid, { ACPTN acptn1, ACPTN acptn2,ACPTN acptn3, ACPTN acptn4 }) 〔SP〕

アラームハンドラ

CRE_ALM(ID almid, { ATR almatr, intptr_t exinf, ALMHDR almhdr }) 〔S〕
AID_ALM(uint_t noalm) 〔SD〕
SAC_ALM(ID almid, { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 }) 〔SP〕

オーバランハンドラ

DEF_OVR({ ATR ovratr, OVRHDR ovrhdr }) 〔S〕

(7) システム状態管理機能

SAC_SYS({ ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 }) 〔SP〕

(8) メモリオブジェクト管理機能

ATT_REG("メモリリージョン名",{ ATR regatr, void *base, SIZE size }) 〔SP〕
DEF_SRG("標準ROMリージョン名", "標準RAMリージョン名") 〔SP〕
ATT_SEC("セクション名", { ATR rematr, "メモリリージョン名" }) 〔SP〕
ATA_SEC("セクション名", { ATR rematr, "メモリリージョン名" }, { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 }) 〔SP〕
LNK_SEC("セクション名", { "メモリリージョン名" }) 〔SP〕
ATT_MOD("オブジェクトモジュール名") 〔SP〕
ATA_MOD("オブジェクトモジュール名", { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 }) 〔SP〕
ATT_MEM({ ATR rematr, void *base, SIZE size }) 〔SP〕
ATA_MEM({ ATR rematr, void *base, SIZE size }, { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 }) 〔SP〕
ATA_PMA({ ATR rematr, void *base, SIZE size, void *paddr }) 〔SP〕
ATA_PMA({ ATR rematr, void *base, SIZE size, void *paddr }, { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 }) 〔SP〕

(9) 割込み管理機能

CFG_INT(INTNO intno, { ATR intatr, PRI intpri }) 〔S〕
CRE_ISR(ID isrid, { ATR isratr, intptr_t exinf, INTNO intno, ISR isr, PRI isrpri }) 〔S〕
ATT_ISR({ ATR isratr, intptr_t exinf, INTNO intno, ISR isr, PRI isrpri }) 〔S〕
AID_ISR(uint_t noisr) 〔SD〕
SAC_ISR(ID isrid, { ACPTN acptn1, ACPTN acptn2, ACPTN acptn3, ACPTN acptn4 }) 〔SP〕
DEF_INH(INHNO inhno, { ATR inhatr, INTHDR inthdr }) 〔S〕

(10) CPU例外管理機能

DEF_EXC(EXCNO excno, { ATR excatr, EXCHDR exchdr }) 〔S〕

(11) 拡張サービスコール管理機能

DEF_SVC(FN fncd, { ATR svcatr, EXTSVC svcrtn, SIZE stksz }) 〔SP〕

(12) システム構成管理機能

LMT_DOM({ PRI mintpri }) 〔SP〕 DEF_ICS({ SIZE istksz, STK_T *istk }) 〔S〕 DEF_STK({ SIZE stksz, STK_T *stk }) 〔S〕 ATT_INI({ ATR iniatr, intptr_t exinf, INIRTN inirtn }) 〔S〕 ATT_TER({ ATR teratr, intptr_t exinf, TERRTN terrtn }) 〔S〕

## 1.3. データ型

### 1.3.1. TOPPERS共通データ型

```
int8_t      符号付き8ビット整数（オプション，C99準拠）
uint8_t     符号無し8ビット整数（オプション，C99準拠）
int16_t     符号付き16ビット整数（C99準拠）
uint16_t    符号無し16ビット整数（C99準拠）
int32_t     符号付き32ビット整数（C99準拠）
uint32_t    符号無し32ビット整数（C99準拠）
int64_t     符号付き64ビット整数（オプション，C99準拠）
uint64_t    符号無し64ビット整数（オプション，C99準拠）
int128_t    符号付き128ビット整数（オプション，C99準拠）
uint128_t   符号無し128ビット整数（オプション，C99準拠）
```

```
int_least8_t    8ビット以上の符号付き整数（C99準拠）
uint_least8_t   int_least8_t型と同じサイズの符号無し整数（C99準拠）
```

```
float32_t   IEEE754準拠の32ビット単精度浮動小数点数（オプション）
double64_t  IEEE754準拠の64ビット倍精度浮動小数点数（オプション）
```

```
bool_t      真偽値（trueまたはfalse）
int_t       16ビット以上の符号付き整数
uint_t      int_t型と同じサイズの符号無し整数
long_t      32ビット以上かつint_t型以上のサイズの符号付き整数
ulong_t     long_t型と同じサイズの符号無し整数
```

```
intptr_t    ポインタを格納できるサイズの符号付き整数（C99準拠）
uintptr_t   intptr_t型と同じサイズの符号無し整数（C99準拠）
```

```
FN          機能コード（符号付き整数，int_tに定義）
ER          エラーコード（符号付き整数，int_tに定義）
ID          オブジェクトのID番号（符号付き整数，int_tに定義）
ATR         オブジェクト属性（符号無し整数，uint_tに定義）
STAT        オブジェクトの状態（符号無し整数，uint_tに定義）
MODE        サービスコールの動作モード（符号無し整数，uint_tに定義）
PRI         優先度（符号付き整数，int_tに定義）
SIZE        メモリ領域のサイズ（符号無し整数，ポインタを格納できる
            サイズの符号無し整数型に定義）
```

```
TMO         タイムアウト指定（符号付き整数，単位はミリ秒，int_tに定義）
RELTIM      相対時間（符号無し整数，単位はミリ秒，uint_tに定義）
SYSTIM      システム時刻（符号無し整数，単位はミリ秒，ulong_tに定義）
SYSUTM      性能評価用システム時刻（符号無し整数，単位はマイクロ秒，
            ulong_tに定義）
```

```
FP          プログラムの起動番地（型の定まらない関数ポインタ）
```

```
ER_BOOL     エラーコードまたは真偽値（符号付き整数，int_tに定義）
ER_ID       エラーコードまたはID番号（符号付き整数，int_tに定義，
            負のID番号は格納できない）
ER_UINT     エラーコードまたは符号無し整数（符号付き整数，int_tに
            定義，符号無し整数を格納する場合の有効ビット数はuint_t
            より1ビット短い）
```

```
MB_T        オブジェクト管理領域を確保するためのデータ型
```

```
ACPTN       アクセス許可パターン（符号無し32ビット整数，uint32_tに
            定義）
```

```
typedef struct acvct {      /* アクセス許可ベクタ */
    ACPTN   acptn1;         /* 通常操作1のアクセス許可パターン */
    ACPTN   acptn2;         /* 通常操作2のアクセス許可パターン */
    ACPTN   acptn3;         /* 管理操作のアクセス許可パターン */
    ACPTN   acptn4;         /* 参照操作のアクセス許可パターン */
} ACVCT;
```

### 1.3.2. カーネルの使用するデータ型

```
TEXPTN      タスク例外要因のビットパターン（符号無し整数，uint_tに定義）
FLGPTN      イベントフラグのビットパターン（符号無し整数，uint_tに定義）
OVRTIM      プロセッサ時間（符号無し整数，単位はマイクロ秒，ulong_tに定義）
INTNO       割込み番号（符号無し整数，uint_tに定義）
INHNO       割込みハンドラ番号（符号無し整数，uint_tに定義）
EXCNO       CPU例外ハンドラ番号（符号無し整数，uint_tに定義）
```

```
TASK        タスクのメインルーチン（関数ポインタ）
TEXRTN      タスク例外処理ルーチン（関数ポインタ）
CYCHDR      周期ハンドラ（関数ポインタ）
ALMHDR      アラームハンドラ（関数ポインタ）
OVRHDR      オーバランハンドラ（関数ポインタ）
ISR         割込みサービスルーチン（関数ポインタ）
INTHDR      割込みハンドラ（関数ポインタ）
EXCHDR      CPU例外ハンドラ（関数ポインタ）
EXTSVC      拡張サービスコール（関数ポインタ）
INIRTN      初期化ルーチン（関数ポインタ）
TERRTN      終了処理ルーチン（関数ポインタ）
```

```
STK_T       スタック領域を確保するためのデータ型
MPF_T       固定長メモリプール領域を確保するためのデータ型
```

メールボックスのメッセージヘッダ【NGKI4001】

```
typedef struct t_msg {
    struct t_msg    *pk_next;
} T_MSG;
```

メールボックスの優先度付きメッセージヘッダ【NGKI4002】

```
typedef struct t_msg_pri {
    T_MSG       msgque;         /* メールボックスのメッセージヘッダ */
    PRI         msgpri;         /* メッセージ優先度 */
} T_MSG_PRI;
```

### 1.3.3. カーネルの使用するパケット形式

(1) タスク管理機能

タスクの生成情報のパケット形式【NGKI4003】

```
typedef struct t_ctsk {
    ATR         tskatr;     /* タスク属性 */
    intptr_t    exinf;      /* タスクの拡張情報 */
    TASK        task;       /* タスクのメインルーチンの先頭番地 */
    PRI         itskpri;    /* タスクの起動時優先度 */
    SIZE        stksz;      /* タスクのスタック領域のサイズ */
    STK_T *     stk;        /* タスクのスタック領域の先頭番地 */
    /* 以下は，保護機能対応カーネルの場合 */
    SIZE        sstksz;     /* タスクのシステムスタック領域のサイズ */
    STK_T *     sstk;       /* タスクのシステムスタック領域の先頭番地 */
} T_CTSK;
```

タスクの現在状態のパケット形式【NGKI4004】

```
typedef struct t_rtsk {
    STAT        tskstat;    /* タスク状態 */
    PRI         tskpri;     /* タスクの現在優先度 */
    PRI         tskbpri;    /* タスクのベース優先度 */
    STAT        tskwait;    /* 待ち要因 */
    ID          wobjid;     /* 待ち対象のオブジェクトのID */
    TMO         lefttmo;    /* タイムアウトするまでの時間 */
    uint_t      actcnt;     /* 起動要求キューイング数 */
    uint_t      wupcnt;     /* 起床要求キューイング数 */
    /* 以下は，保護機能対応カーネルの場合 */
    bool_t      texmsk;     /* タスク例外マスク状態か否か */
    bool_t      waifbd;     /* 待ち禁止状態か否か */
    uint_t      svclevel;   /* 拡張サービスコールのネストレベル */
    /* 以下は，マルチプロセッサ対応カーネルの場合 */
    ID          prcid;      /* 割付けプロセッサのID */
    ID          actprc      /* 次の起動時の割付けプロセッサのID */
} T_RTSK;
```

(2) タスク付属同期機能

```
なし
```

(3) タスク例外処理機能

タスク例外処理ルーチンの定義情報のパケット形式【NGKI4005】

```
typedef struct t_dtex {
    ATR         texatr;     /* タスク例外処理ルーチン属性 */
    TEXRTN      texrtn;     /* タスク例外処理ルーチンの先頭番地 */
} T_DTEX;
```

タスク例外処理の現在状態のパケット形式【NGKI4006】

```
typedef struct t_rtex {
    STAT        texstat;    /* タスク例外処理の状態 */
    TEXPTN      pndptn;     /* 保留例外要因 */
} T_RTEX;
```

(4) 同期・通信機能

セマフォの生成情報のパケット形式【NGKI4007】

```
typedef struct t_csem {
    ATR         sematr;     /* セマフォ属性 */
    uint_t      isemcnt;    /* セマフォの初期資源数 */
    uint_t      maxsem;     /* セマフォの最大資源数 */
} T_CSEM;
```

セマフォの現在状態のパケット形式【NGKI4008】

```
typedef struct t_rsem {
    ID          wtskid;     /* セマフォの待ち行列の先頭のタスクのID番号 */
    uint_t      semcnt;     /* セマフォの資源数 */
} T_RSEM;
```

イベントフラグの生成情報のパケット形式【NGKI4009】

```
typedef struct t_cflg {
    ATR         flgatr;     /* イベントフラグ属性 */
    FLGPTN      iflgptn;    /* イベントフラグの初期ビットパターン */
} T_CFLG;
```

イベントフラグの現在状態のパケット形式【NGKI4010】

```
typedef struct t_rflg {
    ID          wtskid;     /* イベントフラグの待ち行列の先頭のタス
                               クのID番号 */
    FLGPTN      flgptn;     /* イベントフラグのビットパターン */
} T_RFLG;
```

データキューの生成情報のパケット形式【NGKI4011】

```
typedef struct t_cdtq {
    ATR         dtqatr;     /* データキュー属性 */
    uint_t      dtqcnt;     /* データキュー管理領域に格納できるデータ数 */
    void *      dtqmb;      /* データキュー管理領域の先頭番地 */
} T_CDTQ;
```

データキューの現在状態のパケット形式【NGKI4012】

```
typedef struct t_rdtq {
    ID          stskid;     /* データキューの送信待ち行列の先頭のタ
                               スクのID番号 */
    ID          rtskid;     /* データキューの受信待ち行列の先頭のタ
                               スクのID番号 */
    uint_t      sdtqcnt;    /* データキュー管理領域に格納されている
                               データの数 */
} T_RDTQ;
```

優先度データキューの生成情報のパケット形式【NGKI4013】

```
typedef struct t_cpdq {
    ATR         pdqatr;     /* 優先度データキュー属性 */
    uint_t      pdqcnt;     /* 優先度データキュー管理領域に格納でき
                               るデータ数 */
    PRI         maxdpri;    /* 優先度データキューに送信できるデータ
                               優先度の最大値 */
    void *      pdqmb;      /* 優先度データキュー管理領域の先頭番地 */
} T_CPDQ;
```

優先度データキューの現在状態のパケット形式【NGKI4014】

```
typedef struct t_rpdq {
    ID          stskid;     /* 優先度データキューの送信待ち行列の先
                               頭のタスクのID番号 */
    ID          rtskid;     /* 優先度データキューの受信待ち行列の先
                               頭のタスクのID番号 */
    uint_t      spdqcnt;    /* 優先度データキュー管理領域に格納され
                               ているデータの数 */
} T_RPDQ;
```

メールボックスの生成情報のパケット形式【NGKI4015】

```
typedef struct t_cmbx {
    ATR         mbxatr;     /* メールボックス属性 */
    PRI         maxmpri;    /* 優先度メールボックスに送信できるメッ
                               セージ優先度の最大値 */
    void *      mprihd;     /* 優先度別のメッセージキューヘッダ領域
                               の先頭番地 */
} T_CMBX;
```

メールボックスの現在状態のパケット形式【NGKI4016】

```
typedef struct t_rmbx {
    ID          wtskid;     /* メールボックスの待ち行列の先頭のタスク
                               のID番号 */
    T_MSG       *pk_msg;    /* メッセージキューの先頭につながれたメッ
                               セージの先頭番地 */
} T_RMBX;
```

ミューテックスの生成情報のパケット形式【NGKI4017】

```
typedef struct t_cmtx {
    ATR         mtxatr;     /* ミューテックス属性 */
    PRI         ceilpri;    /* ミューテックスの上限優先度 */
} T_CMTX;
```

ミューテックスの現在状態のパケット形式【NGKI4018】

```
typedef struct t_rmtx {
    ID          htskid;     /* ミューテックスをロックしているタス
                               クのID番号 */
    ID          wtskid;     /* ミューテックスの待ち行列の先頭のタ
                               スクのID番号 */
} T_RMTX;
```

メッセージバッファの生成情報のパケット形式【NGKI4037】

```
typedef struct t_cmbf {
    ATR         mbfatr;     /* メッセージバッファ属性 */
    uint_t      maxmsz;     /* メッセージバッファの最大メッセージ
                               サイズ（バイト数）*/
    SIZE        mbfsz;      /* メッセージバッファ管理領域のサイズ
                               （バイト数）*/
    void *      mbfmb;      /* メッセージバッファ管理領域の先頭番地 */
} T_CMBF;
```

メッセージバッファの現在状態のパケット形式【NGKI4038】

```
typedef struct t_rmbf {
    ID      stskid;     /* メッセージバッファの送信待ち行列の先頭の
                           タスクのID番号 */
    ID      rtskid;     /* メッセージバッファの受信待ち行列の先頭の
                           タスクのID番号 */
    uint_t  smbfcnt;    /* メッセージバッファ管理領域に格納されてい
                           るメッセージの数 */
    SIZE    fmbfsz;     /* メッセージバッファ管理領域中の空き領域の
                           サイズ */
} T_RMBF;
```

スピンロックの生成情報のパケット形式【NGKI4019】

```
typedef struct t_cspn {
    ATR         spnatr;     /* スピンロック属性 */
} T_CSPN;
```

スピンロックの現在状態のパケット形式【NGKI4020】

```
typedef struct t_rspn {
    STAT        spnstat     /* スピンロックのロック状態 */
} T_RSPN;
```

(5) メモリプール管理機能

固定長メモリプールの生成情報のパケット形式【NGKI4021】

```
typedef struct t_cmpf {
    ATR         mpfatr;     /* 固定長メモリプール属性 */
    uint_t      blkcnt;     /* 獲得できる固定長メモリブロックの数 */
    uint_t      blksz;      /* 固定長メモリブロックのサイズ */
    MPF_T *     mpf;        /* 固定長メモリプール領域の先頭番地 */
    void *      mpfmb;      /* 固定長メモリプール管理領域の先頭番地 */
} T_CMPF;
```

固定長メモリプールの現在状態のパケット形式【NGKI4022】

```
typedef struct t_rmpf {
    ID          wtskid;     /* 固定長メモリプールの待ち行列の先頭の
                               タスクのID番号 */
    uint_t      fblkcnt;    /* 固定長メモリプール領域の空きメモリ領
                               域に割り付けることができる固定長メモ
                               リブロックの数 */
} T_RMPF;
```

(6) 時間管理機能

周期ハンドラの生成情報のパケット形式【NGKI4023】

```
typedef struct t_ccyc {
    ATR         cycatr;     /* 周期ハンドラ属性 */
    intptr_t    exinf;      /* 周期ハンドラの拡張情報 */
    CYCHDR      cychdr;     /* 周期ハンドラの先頭番地 */
    RELTIM      cyctim;     /* 周期ハンドラの起動周期 */
    RELTIM      cycphs;     /* 周期ハンドラの起動位相 */
} T_CCYC;
```

周期ハンドラの現在状態のパケット形式【NGKI4024】

```
typedef struct t_rcyc {
    STAT        cycstat;    /* 周期ハンドラの動作状態 */
    RELTIM      lefttim;    /* 次に周期ハンドラを起動する時刻までの
                               相対時間 */
    /* 以下は，マルチプロセッサ対応カーネルの場合 */
    ID          prcid;      /* 割付けプロセッサのID */
} T_RCYC;
```

アラームハンドラの生成情報のパケット形式【NGKI4025】

```
typedef struct t_calm {
    ATR         almatr;     /* アラームハンドラ属性 */
    intptr_t    exinf;      /* アラームハンドラの拡張情報 */
    ALMHDR      almhdr;     /* アラームハンドラの先頭番地 */
} T_CALM;
```

アラームハンドラの現在状態のパケット形式【NGKI4026】

```
typedef struct t_ralm {
    STAT        almstat;    /* アラームハンドラの動作状態 */
    RELTIM      lefttim;    /* アラームハンドラを起動する時刻までの
                               相対時間 */
    /* 以下は，マルチプロセッサ対応カーネルの場合 */
    ID          prcid;      /* 割付けプロセッサのID */
} T_RALM;
```

オーバランハンドラの定義情報のパケット形式【NGKI4027】

```
typedef struct t_dovr {
    ATR         ovratr;     /* オーバランハンドラ属性 */
    OVRHDR      ovrhdr;     /* オーバランハンドラの先頭番地 */
} T_DOVR;
```

オーバランハンドラの現在状態のパケット形式【NGKI4028】

```
typedef struct t_rovr {
    STAT    ovrstat;        /* オーバランハンドラの動作状態 */
    OVRTIM  leftotm;        /* 残りプロセッサ時間 */
} T_ROVR;
```

(7) システム状態管理機能

システムの現在状態のパケット形式

未完成

(8) メモリオブジェクト管理機能

メモリオブジェクトの登録情報のパケット形式【NGKI4029】

```
typedef struct t_amem {
    ATR         mematr      /* メモリオブジェクト属性 */
    void *      base        /* 登録するメモリ領域の先頭番地 */
    SIZE        size        /* 登録するメモリ領域のサイズ（バイト数）*/
} T_AMEM;
```

物理メモリ領域の登録情報のパケット形式【NGKI4030】

```
typedef struct t_apma {
    ATR         mematr      /* メモリオブジェクト属性 */
    void *      base        /* 登録するメモリ領域の先頭番地 */
    SIZE        size        /* 登録するメモリ領域のサイズ（バイト数）*/
    void *      paddr       /* 登録するメモリ領域の物理アドレスの先頭
                               番地 */
} T_APMA;
```

メモリオブジェクトの現在状態のパケット形式

未完成

(9) 割込み管理機能

割込み要求ラインの属性の設定情報のパケット形式【NGKI4031】

```
typedef struct t_cint {
    ATR         intatr;     /* 割込み要求ライン属性 */
    PRI         intpri;     /* 割込み優先度 */
} T_CINT;
```

割込みサービスルーチンの生成情報のパケット形式【NGKI4032】

```
typedef struct t_cisr {
    ATR         isratr;     /* 割込みサービスルーチン属性 */
    intptr_t    exinf;      /* 割込みサービスルーチンの拡張情報 */
    INTNO       intno;      /* 割込みサービスルーチンを登録する割込
                               み番号 */
    ISR         isr;        /* 割込みサービスルーチンの先頭番地 */
    PRI         isrpri;     /* 割込みサービスルーチン優先度 */
} T_CISR;
```

割込みサービスルーチンの現在状態のパケット形式

　未完成

割込みハンドラの定義情報のパケット形式【NGKI4033】

```
typedef struct t_dinh {
    ATR         inhatr;     /* 割込みハンドラ属性 */
    INTHDR      inthdr;     /* 割込みハンドラの先頭番地 */
} T_DINH;
```

割込み要求ラインの現在状態のパケット形式

　未完成

(10) CPU例外管理機能

CPU例外ハンドラの定義情報のパケット形式【NGKI4034】

```
typedef struct t_dexc {
    ATR         excatr;     /* CPU例外ハンドラ属性 */
    EXCHDR      exchdr;     /* CPU例外ハンドラの先頭番地 */
} T_DEXC;
```

(11) 拡張サービスコール管理機能

拡張サービスコールの定義情報のパケット形式【NGKI4035】

```
typedef struct t_dsvc {
    ATR         svcatr      /* 拡張サービスコール属性 */
    EXTSVC      svcrtn      /* 拡張サービスコールの先頭番地 */
    SIZE        stksz       /* 拡張サービスコールで使用するスタック
                               サイズ */
} T_DSVC;
```

(12) システム構成管理機能

コンフィギュレーション情報のパケット形式

　未完成

バージョン情報のパケット形式

　未完成

# 1.4. 定数とマクロ

## 1.4.1. TOPPERS共通定数

(1) 一般定数

```
NULL                    無効ポインタ
```

```
true            1       真
false           0       偽
```

```
E_OK            0       正常終了
```

(2) 整数型に格納できる最大値と最小値

```
INT8_MAX            int8_tに格納できる最大値（オプション，C99準拠）
INT8_MIN            int8_tに格納できる最小値（オプション，C99準拠）
UINT8_MAX           uint8_tに格納できる最大値（オプション，C99準拠）
INT16_MAX           int16_tに格納できる最大値（C99準拠）
INT16_MIN           int16_tに格納できる最小値（C99準拠）
UINT16_MAX          uint16_tに格納できる最大値（C99準拠）
INT32_MAX           int32_tに格納できる最大値（C99準拠）
INT32_MIN           int32_tに格納できる最小値（C99準拠）
UINT32_MAX          uint32_tに格納できる最大値（C99準拠）
INT64_MAX           int64_tに格納できる最大値（オプション，C99準拠）
INT64_MIN           int64_tに格納できる最小値（オプション，C99準拠）
UINT64_MAX          uint64_tに格納できる最大値（オプション，C99準拠）
INT128_MAX          int128_tに格納できる最大値（オプション，C99準拠）
INT128_MIN          int128_tに格納できる最小値（オプション，C99準拠）
UINT128_MAX         uint128_tに格納できる最大値（オプション，C99準拠）
```

```
INT_LEAST8_MAX      int_least8_tに格納できる最大値（C99準拠）
INT_LEAST8_MIN      int_least8_tに格納できる最小値（C99準拠）
UINT_LEAST8_MAX     uint_least8_tに格納できる最大値（C99準拠）
INT_MAX             int_tに格納できる最大値（C90準拠）
INT_MIN             int_tに格納できる最小値（C90準拠）
UINT_MAX            uint_tに格納できる最大値（C90準拠）
LONG_MAX            long_tに格納できる最大値（C90準拠）
LONG_MIN            long_tに格納できる最小値（C90準拠）
ULONG_MAX           ulong_tに格納できる最大値（C90準拠）
```

```
FLOAT32_MIN         float32_tに格納できる最小の正規化された正の浮
                    動小数点数（オプション）
FLOAT32_MAX         float32_tに格納できる表現可能な最大の有限浮動
                    小数点数（オプション）
DOUBLE64_MIN        double64_tに格納できる最小の正規化された正の浮
                    動小数点数（オプション）
DOUBLE64_MAX        double64_tに格納できる表現可能な最大の有限浮動
                    小数点数（オプション）
```

(3) 整数型のビット数

```
CHAR_BIT            char型のビット数（C90準拠）
```

(4) オブジェクト属性

```
TA_NULL     0U      オブジェクト属性を指定しない
```

(5) タイムアウト指定

```
TMO_POL     0       ポーリング
TMO_FEVR    -1      永久待ち
TMO_NBLK    -2      ノンブロッキング
```

(6) アクセス許可パターン

```
TACP_KERNEL     0U      カーネルドメインのみにアクセスを許可
TACP_SHARED     ~0U     すべての保護ドメインにアクセスを許可
```

### 1.4.2. TOPPERS共通マクロ

(1) 整数定数を作るマクロ

```
INT8_C(val)         int_least8_t型の定数を作るマクロ（C99準拠）
UINT8_C(val)        uint_least8_t型の定数を作るマクロ（C99準拠）
INT16_C(val)        int16_t型の定数を作るマクロ（C99準拠）
UINT16_C(val)       uint16_t型の定数を作るマクロ（C99準拠）
INT32_C(val)        int32_t型の定数を作るマクロ（C99準拠）
UINT32_C(val)       uint32_t型の定数を作るマクロ（C99準拠）
INT64_C(val)        int64_t型の定数を作るマクロ（オプション，C99準拠）
UINT64_C(val)       uint64_t型の定数を作るマクロ（オプション，C99準拠）
INT128_C(val)       int128_t型の定数を作るマクロ（オプション，C99準拠）
UINT128_C(val)      uint128_t型の定数を作るマクロ（オプション，C99準拠）
```

```
UINT_C(val)         uint_t型の定数を作るマクロ
ULONG_C(val)        ulong_t型の定数を作るマクロ
```

(2) 型に関する情報を取り出すためのマクロ

```
offsetof(structure, field)  構造体structure中のフィールドfieldの
                            バイト位置を返すマクロ（C90準拠）
```

```
alignof(type)               型typeのアラインメント単位を返すマクロ
```

```
ALIGN_TYPE(addr, type)      番地addrが型typeに対してアラインしてい
                            るかどうかを返すマクロ
```

(3) assertマクロ

```
assert(exp)         expが成立しているかを検査するマクロ（C90準拠）
```

(4) コンパイラの拡張機能のためのマクロ

```
inline              インライン関数
Inline              ファイルローカルなインライン関数
asm                 インラインアセンブラ
Asm                 インラインアセンブラ（最適化抑止）
throw()             例外を発生しない関数
NoReturn            リターンしない関数
```

(5) エラーコード生成・分解マクロ

```
ERCD(mercd, sercd)  メインエラーコードmercdとサブエラーコードsercdか
                    ら，エラーコードを生成するためのマクロ
```

```
MERCD(ercd)         エラーコードercdからメインエラーコードを抽出する
                    ためのマクロ
SERCD(ercd)         エラーコードercdからサブエラーコードを抽出するた
                    めのマクロ
```

(6) アクセス許可パターン生成マクロ

```
TACP(domid)         domidで指定される保護ドメインに属する処理単位の
                    みにアクセスを許可するアクセス許可パターン
```

### 1.4.3. カーネル共通定数

(1) オブジェクト属性

```
TA_TPRI         0x01U   タスクの待ち行列をタスクの優先度順に
```

(2) 保護ドメインID

```
TDOM_SELF       0       自タスクの属する保護ドメイン
TDOM_KERNEL     -1      カーネルドメイン
TDOM_NONE       -2      無所属（保護ドメインに属さない）
```

(3) その他のカーネル共通定数

```
TCLS_SELF       0       自タスクの属するクラス
```

```
TPRC_NONE       0       割付けプロセッサの指定がない
TPRC_INI        0       初期割付けプロセッサ
```

```
TSK_SELF        0       自タスク指定
TSK_NONE        0       該当するタスクがない
```

```
TPRI_SELF       0       自タスクのベース優先度の指定
TPRI_INI        0       タスクの起動時優先度の指定
```

```
TIPM_ENAALL     0       割込み優先度マスク全解除
```

### 1.4.4. カーネル共通マクロ

(1) オブジェクト属性を作るマクロ

```
TA_DOM(domid)       domidで指定される保護ドメインに属する
TA_CLS(clsid)       clsidで指定されるクラスに属する
```

(2) サービスコールの呼出し方法を指定するマクロ

```
SVC_CALL(svc)       svcで指定されるサービスコールを関数呼出しによっ
                    て呼び出すための名称
```

### 1.4.5. カーネルの機能毎の定数

(1) タスク管理機能

```
TA_ACT          0x02U   タスクの生成時にタスクを起動する
TA_RSTR         0x04U   生成するタスクを制約タスクとする
TA_FPU                  FPUレジスタをコンテキストに含める
```

```
TTS_RUN         0x01U   実行状態
TTS_RDY         0x02U   実行可能状態
TTS_WAI         0x04U   待ち状態
TTS_SUS         0x08U   強制待ち状態
TTS_WAS         0x0cU   二重待ち状態
TTS_DMT         0x10U   休止状態
```

```
TTW_SLP         0x0001U 起床待ち
TTW_DLY         0x0002U 時間経過待ち
TTW_SEM         0x0004U セマフォの資源獲得待ち
TTW_FLG         0x0008U イベントフラグ待ち
TTW_SDTQ        0x0010U データキューへの送信待ち
TTW_RDTQ        0x0020U データキューからの受信待ち
TTW_SPDQ        0x0100U 優先度データキューへの送信待ち
TTW_RPDQ        0x0200U 優先度データキューからの受信待ち
TTW_MBX         0x0040U メールボックスからの受信待ち
TTW_MTX         0x0080U ミューテックスのロック待ち状態
TTW_SMBF        0x0400U メッセージバッファへの送信待ち
TTW_RMBF        0x0800U メッセージバッファからの受信待ち
TTW_MPF         0x2000U 固定長メモリブロックの獲得待ち
```

TA_FPUの値は，ターゲット定義とする.

(3) タスク例外処理機能

```
TTEX_ENA        0x01U   タスク例外処理許可状態
TTEX_DIS        0x02U   タスク例外処理禁止状態
```

(4) 同期・通信機能

イベントフラグ

```
TA_WMUL         0x02U   複数のタスクが待つのを許す
TA_CLR          0x04U   タスクの待ち解除時にイベントフラグをクリアする
```

```
TWF_ORW         0x01U   イベントフラグのOR待ちモード
TWF_ANDW        0x02U   イベントフラグのAND待ちモード
```

メールボックス

```
TA_MPRI         0x02U   メッセージキューをメッセージの優先度順にする
```

スピンロック

```
TSPN_UNL        0x01U   取得されていない状態
TSPN_LOC        0x02U   取得されている状態
```

(6) 時間管理機能

周期ハンドラ

```
TA_STA          0x02U   周期ハンドラの生成時に周期ハンドラを動作開始する
TA_PHS          0x04U   周期ハンドラを生成した時刻を基準時刻とする
```

```
TCYC_STP        0x01U   周期ハンドラが動作していない状態
TCYC_STA        0x02U   周期ハンドラが動作している状態
```

アラームハンドラ

```
TALM_STP        0x01U   アラームハンドラが動作していない状態
TALM_STA        0x02U   アラームハンドラが動作している状態
```

オーバランハンドラ

```
TOVR_STP        0x01U   オーバランハンドラが動作していない状態
TOVR_STA        0x02U   オーバランハンドラが動作している状態
```

(8) メモリオブジェクト管理機能

```
TA_NOWRITE      0x01U   書込みアクセス禁止
TA_NOREAD       0x02U   読出しアクセス禁止
TA_EXEC         0x04U   実行アクセス許可
TA_MEMINI       0x08U   メモリの初期化を行う
TA_MEMPRSV      0x10U   メモリの初期化を行わない
TA_SDATA        0x20U   ショートデータ領域に配置
TA_UNCACHE      0x40U   キャッシュ禁止
TA_IODEV        0x80U   周辺デバイスの領域
TA_WTHROUGH             ライトスルーキャッシュを用いる
```

```
TPM_WRITE       0x01U   書込みアクセス権のチェック
TPM_READ        0x02U   読出しアクセス権のチェック
TPM_EXEC        0x04U   実行アクセス権のチェック
```

TA_WTHROUGHの値は，ターゲット定義とする．

(9) 割込み管理機能

```
TA_ENAINT       0x01U   割込み要求禁止フラグをクリア
TA_EDGE         0x02U   エッジトリガ
TA_POSEDGE              ポジティブエッジトリガ
TA_NEGEDGE              ネガティブエッジトリガ
TA_BOTHEDGE             両エッジトリガ
TA_LOWLEVEL             ローレベルトリガ
TA_HIGHLEVEL            ハイレベルトリガ
```

```
TA_NONKERNEL    0x02U   カーネル管理外の割込み
```

TA_POSEDGE，TA_NEGEDGE，TA_BOTHEDGE，TA_LOWLEVEL，TA_HIGHLEVELの値は，
ターゲット定義とする．

(10) CPU例外管理機能

```
TA_DIRECT               CPU例外ハンドラを直接呼び出す
```

TA_DIRECTの値は，ターゲット定義とする．

### 1.4.6. カーネルの機能毎のマクロ

(1) タスク管理機能

```
COUNT_STK_T(sz)     サイズszのスタック領域を確保するために必要な
                    STK_T型の配列の要素数
ROUND_STK_T(sz)     要素数COUNT_STK_T(sz)のSTK_T型の配列のサイズ（sz
                    を，STK_T型のサイズの倍数になるように大きい方に
                    丸めた値）
```

(4) 同期・通信機能

```
TSZ_DTQMB(dtqcnt)   dtqcntで指定した数のデータを格納できるデータ
                    キュー管理領域のサイズ（バイト数）
TCNT_DTQMB(dtqcnt)  dtqcntで指定した数のデータを格納できるデータ
                    キュー管理領域を確保するために必要なMB_T型の配
                    列の要素数
```

```
TSZ_PDQMB(pdqcnt)   pdqcntで指定した数のデータを格納できる優先度デー
                    タキュー管理領域のサイズ（バイト数）
TCNT_PDQMB(pdqcnt)  pdqcntで指定した数のデータを格納できる優先度デー
                    タキュー管理領域を確保するために必要なMB_T型の
                    配列の要素数
```

```
TSZ_MBFMB(msgcnt, msgsz)    msgszで指定したサイズのメッセージを，
                            msgcntで指定した数だけ格納できるメッセー
                            ジバッファ管理領域のサイズ（バイト数）
TCNT_MBFMB(msgcnt, msgsz)   msgszで指定したサイズのメッセージを，
                            msgcntで指定した数だけ格納できるメッセー
                            ジバッファ管理領域を確保するために必要
                            なMB_T型の配列の要素数
```

(5) メモリプール管理機能

```
COUNT_MPF_T(blksz)  固定長メモリブロックのサイズがblkszの固定長メモ
                    リプール領域を確保するために，固定長メモリブロッ
                    ク1つあたりに必要なMPF_T型の配列の要素数を求め
                    るマクロ
ROUND_MPF_T(blksz)  要素数COUNT_MPF_T(blksz)のMPF_T型の配列のサイズ
                    （blkszを，MPF_T型のサイズの倍数になるように大き
                    い方に丸めた値）
```

```
TSZ_MPFMB(blkcnt)   blkcntで指定した数の固定長メモリブロックを管理
                    することができる固定長メモリプール管理領域のサ
                    イズ（バイト数）
TCNT_MPFMB(blkcnt)  blkcntで指定した数の固定長メモリブロックを管理
                    することができる固定長メモリプール管理領域を確
                    保するために必要なMB_T型の配列の要素数
```

# 1.5. 構成マクロ

## 1.5.1. TOPPERS共通構成マクロ

(1) 相対時間の範囲

```
TMAX_RELTIM     相対時間に指定できる最大値
```

### 1.5.2. カーネル共通構成マクロ

(1) サポートする機能

```
TOPPERS_SUPPORT_PROTECT          保護機能対応のカーネル
TOPPERS_SUPPORT_MULTI_PRC        マルチプロセッサ対応のカーネル
TOPPERS_SUPPORT_DYNAMIC_CRE      動的生成対応のカーネル
```

(2) 優先度の範囲

```
TMIN_TPRI        タスク優先度の最小値（＝1）
TMAX_TPRI        タスク優先度の最大値
```

(3) プロセッサの数

```
TNUM_PRCID       プロセッサの数
```

(4) 特殊な役割を持ったプロセッサ

```
TOPPERS_MASTER_PRCID         マスタプロセッサのID番号
TOPPERS_SYSTIM_PRCID         システム時刻管理プロセッサのID番号
```

(5) タイマ方式

```
TOPPERS_SYSTIM_LOCAL         ローカルタイマ方式の場合にマクロ定義
TOPPERS_SYSTIM_GLOBAL        グローバルタイマ方式の場合にマクロ定義
```

(6) バージョン情報

```
TKERNEL_MAKER        カーネルのメーカコード（＝0x0118）
TKERNEL_PRID         カーネルの識別番号
TKERNEL_SPVER        カーネル仕様のバージョン番号
TKERNEL_PRVER        カーネルのバージョン番号
```

### 1.5.3. カーネルの機能毎の構成マクロ

(1) タスク管理機能

```
TMAX_ACTCNT      タスクの起動要求キューイング数の最大値
```

| | |
|---|---|
| TNUM_TSKID | 登録できるタスクの数（動的生成対応でないカーネルでは，静的APIによって登録されたタスクの数に一致） |

(2) タスク付属同期機能

| | |
|---|---|
| TMAX_WUPCNT | タスクの起床要求キューイング数の最大値 |

(3) タスク例外処理機能

| | |
|---|---|
| TBIT_TEXPTN | タスク例外要因のビット数（TEXPTNの有効ビット数） |

(4) 同期・通信機能

セマフォ

| | |
|---|---|
| TMAX_MAXSEM | セマフォの最大資源数の最大値 |

| | |
|---|---|
| TNUM_SEMID | 登録できるセマフォの数（動的生成対応でないカーネルでは，静的APIによって登録されたセマフォの数に一致） |

イベントフラグ

| | |
|---|---|
| TBIT_FLGPTN | イベントフラグのビット数（FLGPTNの有効ビット数） |

| | |
|---|---|
| TNUM_FLGID | 登録できるイベントフラグの数（動的生成対応でないカーネルでは，静的APIによって登録されたイベントフラグの数に一致） |

データキュー

| | |
|---|---|
| TNUM_DTQID | 登録できるデータキューの数（動的生成対応でないカーネルでは，静的APIによって登録されたデータキューの数に一致） |

優先度データキュー

```
TMIN_DPRI       データ優先度の最小値（＝1）
TMAX_DPRI       データ優先度の最大値
```

```
TNUM_PDQID      登録できる優先度データキューの数（動的生成対応でない
                カーネルでは，静的APIによって登録された優先度データ
                キューの数に一致）
```

メールボックス

```
TMIN_MPRI       メッセージ優先度の最小値（＝1）
TMAX_MPRI       メッセージ優先度の最大値
```

```
TNUM_MBXID      登録できるメールボックスの数（動的生成対応でないカー
                ネルでは，静的APIによって登録されたメールボックスの
                数に一致）
```

ミューテックス

```
TNUM_MTXID      登録できるミューテックスの数（動的生成対応でないカー
                ネルでは，静的APIによって登録されたミューテックスの
                数に一致）
```

メッセージバッファ

```
TNUM_MBFID      登録できるメッセージバッファの数（動的生成対応でない
                カーネルでは，静的APIによって登録されたメッセージバ
                ッファの数に一致）
```

スピンロック

```
TNUM_SPNID      登録できるスピンロックの数（動的生成対応でないカー
                ネルでは，静的APIによって登録されたミューテックスの
                数に一致）
```

(5) メモリプール管理機能

固定長メモリプール

```
TNUM_MPFID      登録できる固定長メモリプールの数（動的生成対応でない
                カーネルでは，静的APIによって登録された固定長メモリ
                プールの数に一致）
```

(6) 時間管理機能

システム時刻管理

```
TIC_NUME    タイムティックの周期（単位はミリ秒）の分子
TIC_DENO    タイムティックの周期（単位はミリ秒）の分母
```

```
TOPPERS_SUPPORT_GET_UTM     get_utmがサポートされている
```

周期ハンドラ

```
TNUM_CYCID      登録できる周期ハンドラの数（動的生成対応でないカー
                ネルでは，静的APIによって登録された周期ハンドラの数
                に一致）
```

アラームハンドラ

```
TNUM_ALMID      登録できるアラームハンドラの数（動的生成対応でない
                カーネルでは，静的APIによって登録されたアラームハン
                ドラの数に一致）
```

オーバランハンドラ

```
TMAX_OVRTIM     プロセッサ時間に指定できる最大値
```

```
TOPPERS_SUPPORT_OVRHDR      オーバランハンドラ機能がサポートされて
                            いる
```

(7) システム状態管理機能

```
なし
```

(8) メモリオブジェクト管理機能

```
TOPPERS_SUPPORT_ATT_MOD     ATT_MOD／ATA_MODがサポートされている
TOPPERS_SUPPORT_ATT_PMA     ATT_PMA／ATA_PMA／att_pmaがサポートさ
                            れている
```

(9) 割込み管理機能

```
TMIN_INTPRI     割込み優先度の最小値（最高値）
TMAX_INTPRI     割込み優先度の最大値（最低値，＝-1）
```

```
TMIN_ISRPRI     割込みサービスルーチン優先度の最小値（＝1）
TMAX_ISRPRI     割込みサービスルーチン優先度の最大値
```

```
TOPPERS_SUPPORT_DIS_INT     dis_intがサポートされている
TOPPERS_SUPPORT_ENA_INT     ena_intがサポートされている
```

(10) CPU例外管理機能

```
なし
```

(11) 拡張サービスコール管理機能

```
TNUM_FNCD       登録できる拡張サービスコールの数（動的生成対応でな
                いカーネルでは，静的APIによって登録された拡張サービ
                スコールの数に一致）
```

(12) システム構成管理機能

```
なし
```

## 1.6. エラーコード一覧

(1) メインエラーコード

```
E_SYS       -5      システムエラー
E_NOSPT     -9      未サポート機能
E_RSFN      -10     予約機能コード
E_RSATR     -11     予約属性
E_PAR       -17     パラメータエラー
E_ID        -18     不正ID番号
E_CTX       -25     コンテキストエラー
E_MACV      -26     メモリアクセス違反
E_OACV      -27     オブジェクトアクセス違反
E_ILUSE     -28     サービスコール不正使用
E_NOMEM     -33     メモリ不足
E_NOID      -34     ID番号不足
E_NORES     -35     資源不足
E_OBJ       -41     オブジェクト状態エラー
E_NOEXS     -42     オブジェクト未登録
E_QOVR      -43     キューイングオーバフロー
E_RLWAI     -49     待ち禁止状態または待ち状態の強制解除
E_TMOUT     -50     ポーリング失敗またはタイムアウト
E_DLT       -51     待ちオブジェクトの削除または再初期化
E_CLS       -52     待ちオブジェクトの状態変化
E_WBLK      -57     ノンブロッキング受付け
E_BOVR      -58     バッファオーバフロー
```

## 1.7. 機能コード一覧【NGKI4036】

| | -0 | -1 | -2 | -3 |
|---|---|---|---|---|
| -0x01 | 予約 | 予約 | 予約 | 予約 |
| -0x05 | act_tsk | iact_tsk | can_act | ext_tsk |
| -0x09 | ter_tsk | chg_pri | get_pri | get_inf |
| -0x0d | slp_tsk | tslp_tsk | wup_tsk | iwup_tsk |
| -0x11 | can_wup | rel_wai | irel_wai | 予約 |
| -0x15 | dis_wai | idis_wai | ena_wai | iena_wai |
| -0x19 | sus_tsk | rsm_tsk | dly_tsk | 予約 |
| -0x1d | ras_tex | iras_tex | dis_tex | ena_tex |
| -0x21 | sns_tex | ref_tex | 予約 | 予約 |
| -0x25 | sig_sem | isig_sem | wai_sem | pol_sem |
| -0x29 | twai_sem | 予約 | 予約 | 予約 |
| -0x2d | set_flg | iset_flg | clr_flg | wai_flg |

| | | | | |
|---|---|---|---|---|
| -0x31 | pol_flg | twai_flg | 予約 | 予約 |
| -0x35 | snd_dtq | psnd_dtq | ipsnd_dtq | tsnd_dtq |
| -0x39 | fsnd_dtq | ifsnd_dtq | rcv_dtq | prcv_dtq |
| -0x3d | trcv_dtq | 予約 | 予約 | 予約 |
| -0x41 | snd_pdq | psnd_pdq | ipsnd_pdq | tsnd_pdq |
| -0x45 | rcv_pdq | prcv_pdq | trcv_pdq | 予約 |
| -0x49 | snd_mbx | rcv_mbx | prcv_mbx | trcv_mbx |
| -0x4d | loc_mtx | ploc_mtx | tloc_mtx | unl_mtx |
| -0x51 | snd_mbf | psnd_mbf | tsnd_mbf | rcv_mbf |
| -0x55 | prcv_mbf | trcv_mbf | 予約 | 予約 |
| -0x59 | get_mpf | pget_mpf | tget_mpf | rel_mpf |
| -0x5d | get_tim | get_utm | 予約 | ref_ovr |
| -0x61 | sta_cyc | stp_cyc | 予約 | 予約 |
| -0x65 | sta_alm | ista_alm | stp_alm | istp_alm |
| -0x69 | sta_ovr | ista_ovr | stp_ovr | istp_ovr |
| -0x6d | sac_sys | ref_sys | rot_rdq | irot_rdq |
| -0x71 | get_did | 予約 | get_tid | iget_tid |
| -0x75 | loc_cpu | iloc_cpu | unl_cpu | iunl_cpu |
| -0x79 | dis_dsp | ena_dsp | sns_ctx | sns_loc |
| -0x7d | sns_dsp | sns_dpn | sns_ker | ext_ker |
| -0x81 | att_mem | det_mem | sac_mem | prb_mem |
| -0x85 | ref_mem | 予約 | att_pma | 予約 |
| -0x89 | cfg_int | dis_int | ena_int | ref_int |
| -0x8d | chg_ipm | get_ipm | 予約 | 予約 |
| -0x91 | xsns_dpn | xsns_xpn | 予約 | 予約 |
| -0x95 | ref_cfg | ref_ver | 予約 | 予約 |
| -0x99 | 予約 | 予約 | 予約 | 予約 |
| -0x9d | 予約 | 予約 | 予約 | 予約 |
| -0xa1 | 予約 | ini_sem | ini_flg | ini_dtq |
| -0xa5 | ini_pdq | ini_mbx | ini_mtx | ini_mbf |

| | | | | |
|---|---|---|---|---|
| -0xa9 | ini_mpf | 予約 | 予約 | 予約 |
| -0xad | 予約 | 予約 | 予約 | 予約 |
| -0xb1 | ref_tsk | ref_sem | ref_flg | ref_dtq |
| -0xb5 | ref_pdq | ref_mbx | ref_mtx | ref_mbf |
| -0xb9 | ref_mpf | ref_cyc | ref_alm | ref_isr |
| -0xbd | ref_spn | 予約 | 予約 | 予約 |
| -0xc1 | acre_tsk | acre_sem | acre_flg | acre_dtq |
| -0xc5 | acre_pdq | acre_mbx | acre_mtx | acre_mbf |
| -0xc9 | acre_mpf | acre_cyc | acre_alm | acre_isr |
| -0xcd | acre_spn | 予約 | 予約 | 予約 |
| -0xd1 | del_tsk | del_sem | del_flg | del_dtq |
| -0xd5 | del_pdq | del_mbx | del_mtx | del_mbf |
| -0xd9 | del_mpf | del_cyc | del_alm | del_isr |
| -0xdd | del_spn | 予約 | 予約 | 予約 |
| -0xe1 | sac_tsk | sac_sem | sac_flg | sac_dtq |
| -0xe5 | sac_pdq | 予約 | sac_mtx | sac_mbf |
| -0xe9 | sac_mpf | sac_cyc | sac_alm | sac_isr |
| -0xed | sac_spn | 予約 | 予約 | 予約 |
| -0xf1 | def_tex | def_ovr | def_inh | def_exc |
| -0xf5 | def_svc | 予約 | 予約 | 予約 |
| -0xf9 | 予約 | 予約 | 予約 | 予約 |
| -0xfd | 予約 | 予約 | 予約 | 予約 |
| -0x101 | mact_tsk | imact_tsk | mig_tsk | 予約 |
| -0x105 | msta_cyc | 予約 | msta_alm | imsta_alm |
| -0x109 | mrot_rdq | imrot_rdq | get_pid | iget_pid |
| -0x10d | 予約 | 予約 | 予約 | 予約 |
| -0x111 | loc_spn | iloc_spn | try_spn | itry_spn |
| -0x115 | unl_spn | iunl_spn | 予約 | 予約 |
| -0x119 | 予約 | 予約 | 予約 | 予約 |
| -0x11d | 予約 | 予約 | 予約 | 予約 |

【μITRON4.0仕様との関係】

サービスコールの機能コードを割り当てなおした.

## 1.8. カーネルオブジェクトに対するアクセスの種別

| オブジェクトの種類 | 通常操作1 | 通常操作2 | 管理操作 | 参照操作 |
|---|---|---|---|---|
| メモリオブジェクト | 書込み | 読出し | det_mem | ref_mem |
| | | 実行 | sac_mem | prb_mem |
| タスク | act_tsk | ter_tsk | del_tsk | get_pri |
| | mact_tsk | chg_pri | sac_tsk | ref_tsk |
| | can_act | rel_wai | def_tex | ref_tex |
| | mig_tsk | sus_tsk | | ref_ovr |
| | wup_tsk | rsm_tsk | | prb_mem |
| | can_wup | dis_wai | | |
| | | ena_wai | | |
| | | ras_tex | | |
| | | sta_ovr | | |
| | | stp_ovr | | |
| セマフォ | sig_sem | wai_sem | del_sem | ref_sem |
| | | pol_sem | ini_sem | |
| | | twai_sem | sac_sem | |
| イベントフラグ | set_flg | wai_flg | del_flg | ref_flg |
| | clr_flg | pol_flg | ini_flg | |
| | | twai_flg | sac_flg | |
| データキュー | snd_dtq | rcv_dtq | del_dtq | ref_dtq |
| | psnd_dtq | prcv_dtq | ini_dtq | |
| | tsnd_dtq | trcv_dtq | sac_dtq | |
| | fsnd_dtq | | | |
| 優先度データキュー | snd_pdq | rcv_pdq | del_pdq | ref_pdq |
| | psnd_pdq | prcv_pdq | ini_pdq | |
| | tsnd_pdq | trcv_pdq | sac_pdq | |

| オブジェクトの種類 | 通常操作1 | 通常操作2 | 管理操作 | 参照操作 |
|---|---|---|---|---|
| メッセージバッファ | snd_mbf | rcv_mbf | del_mbf | ref_mbf |
| | psnd_mbf | prcv_mbf | ini_mbf | |
| | tsnd_mbf | trcv_mbf | sac_mbf | |
| ミューテックス | loc_mtx | | del_mtx | ref_mtx |
| | ploc_mtx | | ini_mtx | |
| | tloc_mtx | | sac_mtx | |
| | unl_mtx | | | |
| スピンロック | loc_spn | | del_spn | ref_spn |
| | try_spn | | sac_spn | |
| | unl_spn | | | |
| 固定長メモリプール | get_mpf | rel_mpf | del_mpf | ref_mpf |
| | pget_mpf | | ini_mpf | |
| | tget_mpf | | sac_mpf | |
| 周期ハンドラ | sta_cyc | stp_cyc | del_cyc | ref_cyc |
| | msta_cyc | | sac_cyc | |
| アラームハンドラ | sta_alm | stp_alm | del_alm | ref_alm |
| | msta_alm | | sac_alm | |
| 割込みサービスルーチン | | | del_isr | ref_isr |
| | | | sac_isr | |
| システム状態 | rot_rdq | loc_cpu | acre_yyy | get_tim |
| | mrot_rdq | unl_cpu | att_mem | get_ipm |
| | dis_dsp | dis_int | att_pma | ref_sys |
| | ena_dsp | ena_int | cfg_int | ref_int |
| | | chg_ipm | def_inh | ref_cfg |
| | | | def_exc | ref_ver |
| | def_svc | | def_ovr | |

すべての保護ドメインから呼び出すことができるサービスコール：

- 自タスクへの操作（ext_tsk，get_inf，slp_tsk，tslp_tsk，dly_tsk，dis_tex，ena_tex）
- タスク例外状態参照（sns_tex）

- 性能評価用システム時刻の参照（get_utm）
- システム状態参照（get_tid，get_did，get_pid，sns_ctx，sns_loc，sns_dsp，sns_dpn，sns_ker）
- CPU例外発生時の状態参照（xsns_dpn，xsns_xpn）
- 拡張サービスコールの呼出し（cal_svc）

カーネルドメインのみから呼び出すことができるサービスコール：

- システム状態のアクセス許可ベクタの設定（sac_sys）
- カーネルの終了（ext_ker）
- 非タスクコンテキスト専用のサービスコール

【補足説明】

xsns_dpnとxsns_xpnは，エラーコードを返さないために，すべての保護ドメインから呼び出すことができるサービスコールとしているが，タスクコンテキストから呼び出した場合には必ずtrueが返ることとしており，実質的にはカーネルドメインのみから呼び出すことができる．

【μITRON4.0/PX仕様との関係】

get_priは，μITRON4.0/PX仕様ではタスクに対する通常操作1としていたのを，タスクに対する参照操作に変更した．また，get_ipm（μITRON4.0/PX仕様ではget_ixx）をシステム状態に対する通常操作2から参照操作に，sac_sysをシステム状態に対する管理操作からカーネルドメインのみから呼び出すことができるサービスコールに変更した．システム時刻に対するアクセス許可ベクタは廃止し，get_timはシステム状態に対する参照操作とした．

【仕様変更の経緯】

この仕様のRelease 1.5以前では，unl_mtxは，アクセス許可ベクタによるアクセス保護を行わないサービスコールとしていた．これは，ミューテックスをロックしたタスク以外がunl_mtxを呼び出すとE_ILUSEエラーとなるため，実質的には対象ミューテックスの通常操作1としてアクセス保護されているとみなすことができると考えたためである．しかし，タスクが拡張サービスコールの中でミューテックスをロックした場合，アクセス許可ベクタではアクセスが許可されていないミューテックスをロックすることができる．このようなミューテックスのロック解除は，タスクから直接unl_mtxを呼んで行うのではなく，拡張サービスコールの中で行うべきと考えられる．そこで，unl_mtxを，対象ミューテックスの通常操作1としてアクセス保護する仕様に変更した．なお，HRP2カーネル Release 2.1以前のバージョンは，古い仕様に従って実装されている．

# 1.9. ターゲット定義事項一覧

・割込み優先度の段階数［NGKI0256］

・割込み番号の付与方法［NGKI0272］

・割込みハンドラ番号の付与方法［NGKI0273］

・割込み番号に対応しない割込みハンドラ番号や，割込みハンドラ番号に対応しない割込み番号を設けるか［NGKI0276］

・受け付けた割込み要求に対して，割込みサービスルーチンも割込みハンドラも登録していない場合の振舞い［NGKI0249］

・割込み要求禁止フラグがサポートされているか［NGKI0260］［NGKI0261］

・割込み要求禁止フラグの振舞いを仕様と異なるものとするか［NGKI0261］

・割込み要求ラインのトリガモードの設定がサポートされているか［NGKI0267］

・割込み要求ラインをエッジトリガに設定する場合に，ポジティブエッジトリガかネガティブエッジトリガか両エッジトリガかを設定できるか［NGKI0265］

・割込み要求ラインをレベルトリガに設定する場合に，ローレベルトリガかハイレベルトリガかを設定できるか［NGKI0266］

・あるプロセッサで割込み要求禁止フラグを動的にセット／クリアしても，他のプロセッサに対しては割込みがマスク／マスク解除されないものとするか　〔M〕［NGKI0281］

・TMIN_INTPRIを固定するか設定できるようにするかと，設定できるようにする場合の設定方法［NGKI0288］

・NMI以外にカーネル管理外の割込みを設けるか（設けられるようにするか）　［NGKI0289］

・カーネル管理外の割込みハンドラが実行開始される時のシステム状態とコンテキスト，割込みハンドラの終了時に行われる処理，割込みハンドラの記述　方法［NGKI0292］

・カーネル管理外の割込みの設定方法として，3つの方法のいずれを採用するか　［NGKI0295］

・カーネル管理外とされた割込みに対して，カーネルのAPIにより割込みハンドラを登録できるかと，割込み要求ラインの属性を設定できるか［NGKI0297］

・CPU例外ハンドラ番号の付与方法［NGKI0306］

・発生したCPU例外に対して，CPU例外ハンドラを登録していない場合の振舞い　［NGKI0314］

・メモリオブジェクトの先頭番地とサイズに対する制約〔P〕［NGKI0070］　［NGKI2774］

・コンパイラが出力しないセクションの中で，どれを標準のセクションと扱う　か〔P〕［NGKI0113］

・保護ドメイン毎の標準セクションのセクション名を，標準のセクション名と　保護ドメイン名を"_"でつないだものとする仕様を変更するか〔P〕［NGKI0116］

・タスクのユーザスタック領域はそのタスク（とカーネルドメインに属する処理単位）のみがアクセスできるという仕様を変更するか〔P〕［NGKI0074］

・メモリオブジェクトに対して，通常のメモリアクセスにより，許可されていない書込みアクセスまたは読出しアクセス（実行アクセスを含む）を行おうとした場合に，どのCPU例外ハンドラが起動されるか〔P〕［NGKI0411］

・メモリオブジェクトに対して，サービスコールを通じて，許可されていない書込みアクセスまたは読出しアクセスを行おうとした場合に，サービスコールからE_MACVエラーが返るか，メモリアクセス違反ハンドラが起動されるか　〔P〕［NGKI0413］

・メモリアクセス違反ハンドラで，アクセス違反を発生させたアクセスに関する情報（アクセスした番地，アクセスの種別，アクセスした命令の番地など）を参照する方法〔P〕［NGKI0414］

・メモリオブジェクトの書込みアクセスと読出しアクセス（実行アクセスを含む）に対して設定できるアクセス許可パターンに対する制限〔P〕［NGKI0417］

・1つの保護ドメインに登録できるメモリオブジェクトの数に対する制限〔P〕　［NGKI0423］

・ユーザスタック領域に対して実行アクセスを行えるか〔P〕［NGKI0440］

・タスクのユーザスタック領域を，そのタスクが属する保護ドメイン全体からアクセスできるものとするか〔P〕［NGKI0441］

・使用できるクラスのID番号とその属性〔M〕［NGKI0107］

・どのプロセッサをマスタプロセッサとするか〔M〕［NGKI0101］

・ローカルタイマ方式とグローバルタイマ方式のどちらの方式を用いることができるか〔M〕［NGKI0108］

・グローバルタイマ方式の場合に，どのプロセッサをシステム時刻管理プロセッサとするか〔M〕［NGKI0111］

・int8_t，uint8_t，int64_t，uint64_t，int128_t，uint128_t，float32_t，double64_tが使用できるか［NGKI0488］［NGKI0490］

・ターゲット定義のタスク属性［NGKI1016］

・タスクが用いるスタック領域のサイズの最小値［NGKI1042］

・タスクのシステムスタック領域のサイズの最小値〔P〕［NGKI1044］

・タスクが用いるスタック領域の先頭番地とサイズに対する制約［NGKI1050］　［NGKI1056］

・ユーザスタックのスタック領域（ユーザスタック領域）をアプリケーションで確保する方法〔P〕［NGKI1059］

・タスクのシステムスタック領域の先頭番地とサイズに対する制約〔P〕　［NGKI1062］［NGKI1065］［NGKI1070］

・データキュー管理領域の先頭番地に対する制約［NGKI1687］

・優先度データキュー管理領域の先頭番地に対する制約［NGKI1824］

・メッセージバッファ管理領域の先頭番地とサイズに対する制約［NGKI3319］　［NGKI3324］

・生成できるスピンロックの数の上限〔M〕［NGKI2142］

・スピンロックに対して，複数のプロセッサがロックの取得を待っている時に，
どのプロセッサが最初にロックを取得できるか〔M〕［NGKI2183］

・固定長メモリプール領域の先頭番地に対する制約［NGKI2249］

・固定長メモリプール管理領域の先頭番地に対する制約［NGKI2256］

・タイムティックの周期［NGKI2335］

・マルチプロセッサ対応カーネルにおける性能評価用システム時刻の扱い〔M〕　［NGKI2346］

・get_utmがサポートされているか［NGKI2360］

・オーバランハンドラ機能がサポートされているか［NGKI2598］

・オーバランハンドラ機能のプロセッサ時間に指定できる値の上限［NGKI2594］

・ターゲット定義のメモリリージョン属性〔P〕

・メモリリージョンの先頭番地とサイズに対する制約〔P〕［NGKI2768］

・メモリオブジェクトに対するTA_NOWRITE属性，TA_NOREAD属性，TA_EXEC属性
の内，どのような場合にどの属性の指定が無視されるか〔P〕［NGKI2782］

・ショートデータ領域がサポートされておらず，TA_SDATA属性が無視されるか　〔P〕［NGKI2789］

・TA_NOWRITEを指定した場合に，TA_SDATAが無視されるか〔P〕［NGKI2790］

・TA_UNCACHE属性やTA_IODEV属性を指定しても意味がなく，これらの属性が無　視されるか〔P〕［NGKI2792］

・キャッシュ禁止にできないメモリオブジェクトと周辺デバイスの領域として
扱うことができないメモリオブジェクト〔P〕［NGKI2793］

・ターゲット定義のメモリオブジェクト属性〔P〕［NGKI2794］

・ATA_SECにより登録できるセクションが属する保護ドメインや登録できる数　に対する制限〔P〕［NGKI2831］

・ATT_MOD／ATA_MODがサポートされているか〔P〕［NGKI2859］

・ATT_MOD／ATA_MODにより登録されるセクション毎のメモリオブジェクトに設

定されるメモリオブジェクト属性〔P〕［NGKI2850］

・クラスの囲みの中に記述されたATT_MOD／ATA_MODにおいて，クラスの標準メモリリージョンが定義されている場合でも，共通の標準メモリリージョンに配置されるセクション〔PM〕［NGKI3271］

・ATA_MODにより登録できるオブジェクトモジュールが属する保護ドメインや登録できる数に対する制限〔P〕［NGKI2857］

・ATT_MEM／ATA_MEMにより登録できるメモリオブジェクトが属する保護ドメインや登録できる数に対する制限〔P〕［NGKI2878］

・ATT_MEM／ATA_MEM／att_memにより登録するメモリ領域の先頭番地とサイズに対する制約〔P〕［NGKI2880］

・ATT_PMA／ATA_PMA／att_pmaがサポートされているか〔P〕［NGKI2903］［HRPS0156］

・ATT_PMA／ATA_PMAにより登録できるメモリオブジェクトが属する保護ドメインや登録できる数に対する制限〔P〕［NGKI2898］

・ATT_PMA／ATA_PMA／att_pmaにより登録するメモリ領域の先頭番地とサイズ，物理アドレス空間における先頭番地に対する制約〔P〕［NGKI2900］

・ターゲット定義の割込み要求ライン属性［NGKI2945］

・割込みハンドラ属性にTA_NONKERNELを指定できるか［NGKI2957］

・その他のターゲット定義の割込みハンドラ属性［NGKI2959］

・cfg_intにおいて，複数の割込み要求ラインの割込み優先度が連動して設定されるか〔D〕［NGKI2980］

・CFG_INT／cfg_intで，カーネル管理外の割込み要求ラインに対しても属性を設定できるか［NGKI2982］

・CFG_INT／cfg_intで，各割込み要求ラインに対して設定できる割込み要求ライン属性／割込み優先度に対する制限［NGKI2986］

・割込みサービスルーチンが属することができるクラスに対する制限〔M〕［NGKI3018］

・CRE_ISR／ATT_ISRにおいて，isrが不正である場合にE_PARエラーが検出されるか［NGKI3020］

・DEF_INH／def_inhで，カーネル管理外の割込みに対しても割込みハンドラを定義できるか［NGKI3064］

・カーネル管理外に固定されている割込みハンドラがあるか［NGKI3067］

・カーネル管理に固定されている割込みハンドラがあるか［NGKI3068］

・割込みハンドラが属することができるクラスに対する制限〔M〕［NGKI3074］

・def_inhで，静的APIで定義された割込みハンドラの定義を解除できるか〔D〕　［NGKI3077］

・DEF_INH／def_inhで割込みハンドラを定義（または定義解除）できない割込みハンドラ番号［NGKI3078］

・def_inhを呼び出したタスクが割り付けられているプロセッサから定義（または定義解除）できない割込みハンドラ〔M〕［NGKI3079］

・DEF_INHにおいて，inthdrが不正である場合にE_PARエラーが検出されるか　［NGKI3080］

・dis_intがサポートされているか［NGKI3091］

・dis_intにより，どのような場合に割込み要求ラインの割込み要求禁止フラグをセットできないか［NGKI3087］

・dis_intにおいて，割込み要求禁止フラグの振舞いが，この仕様の規定と異な　るか［NGKI3089］

・ena_intがサポートされているか［NGKI3104］

・ena_intにより，どのような場合に割込み要求ラインの割込み要求禁止フラグをクリアできないか［NGKI3100］

・ena_intにおいて，割込み要求禁止フラグの振舞いが，この仕様の規定と異な　るか［NGKI3102］

・chg_ipmにより，割込み優先度マスクをTMIN_INTPRIよりも小さい値に変更で　きるか［NGKI3114］

・ターゲット定義のCPU例外ハンドラ属性［NGKI3123］

・def_excで，静的APIで定義されたCPU例外ハンドラの定義を解除できるか〔D〕　［NGKI3148］

・DEF_EXCにおいて，exchdrが不正である場合にE_PARエラーが検出されるか　［NGKI3149］

・非タスクコンテキスト用スタック領域のサイズの最小値［NGKI3254］

・非タスクコンテキスト用スタック領域の先頭番地とサイズに対する制約　［NGKI3220］［NGKI3222］

・DEF_ICSにより非タスクコンテキスト用スタック領域を設定しない場合の，非タスクコンテキスト用スタック領域のデフォルトのサイズ［NGKI3224］

・共有スタック領域のサイズの最小値［NGKI3255］

・共有スタック領域の先頭番地とサイズに対する制約［NGKI3234］［NGKI3236］

・ATT_INIにおいて，inirtnが不正である場合にE_PARエラーが検出されるか　［NGKI3246］

・ATT_TERにおいて，terrtnが不正である場合にE_PARエラーが検出されるか　［NGKI3253］

## 1.10. 省略名の元になった英語

### 1.10.1. サービスコールと静的APIの名称の中のxxxの元になった英語

```
xxx     元になった英語
----------------------
act     activate
aid     automatically assigned ID
ata     attach with access control vector
att     attach
cal     call
can     cancel
cfg     configure
chg     change
clr     clear
cre     create
def     define
del     delete
det     detach
dis     disable
dly     delay
ena     enable
epr     execution priority
ext     exit
get     get
ini     initialize
lmt     limit
lnk     link
loc     lock
mig     migrate
pol     poll
prb     probe
ras     raise
rcv     receive
ref     reference
rel     release
rot     rotate
rsm     resume
sac     set access control vector
set     set
sig     signal
slp     sleep
snd     send
sns     sense
sta     start
stp     stop
```

```
sus     suspend
ter     terminate
try     try
unl     unlock
wai     wait
wup     wake up
```

### 1.10.2. サービスコールと静的APIの名称の中のyyy の元になった英語

```
yyy     元になった英語
----------------------
act     activation
alm     alarm handler
cfg     configuration
cpu     CPU
ctx     context
cyc     cyclic handler
did     domain ID
dom     domain
dpn     dispatch pending
dsp     dispatch
dtq     data queue
exc     exception
flg     eventflag
ics     interrupt context stack
inf     information
inh     interrupt handler
ini     initilization
int     interrupt
ipm     interrupt priority mask
isr     interrupt service routine
ker     kernel
loc     lock
mbf     message buffer
mbx     mailbox
mpf     fixed-sized memory pool
mem     memory
mod     module
mtx     mutex
ovr     overrun handler
pdq     priority data queue
pid     processor ID
pma     physical memory area
pri     priority
rdq     ready queue
reg     region
```

```
sec     section
sem     semaphore
srg     standard memory region
spn     spin lock
stk     stack
sys     system
svc     service call
ter     termination
tex     task exception
tid     task ID
tim     time
tsk     task
utm     time in micro second
ver     version
wai     wait
wup     wake up
xpn     exception pending
```

### 1.10.3. サービスコールの名称の中のzの元になった英語

```
z       元になった英語
----------------------
a       automatic ID assignment
f       force
i       interrupt
m       multiprocessor
p       poll
t       timeout
x       exception
```

## 1.11. バージョン履歴

```
2008年11月19日 Release 1.0.0       最初のリリース
2009年5月8日   Release 1.1.0       FMPカーネルに関する記述が完成
2010年5月10日  Release 1.2.0
2011年5月5日   Release 1.3.0       HRP2カーネルに関する記述が完成
2012年5月16日  Release 1.4.0       SSPカーネルに関する記述が完成
2012年12月19日 Release 1.5.0       HRP2カーネルの仕様変更を反映
2014年1月16日  Release 1.6.0
2014年11月17日 Release 1.7.0
2015年5月30日  Release 1.7.1
```

以上