### **AArch64.S1TTWParamsEL10**

****// AArch64.S1TTWParamsEL10()

// =========================

// Gather stage 1 translation table walk parameters for EL1&0 regime

// (with EL2 enabled or disabled)

S1TTWParams AArch64.S1TTWParamsEL10([VARange](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Translation?lang=en#VARange) varange)

[S1TTWParams](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Translation?lang=en#S1TTWParams) walkparams;

if varange == [VARange\_LOWER](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Translation?lang=en#VARange_LOWER) then

walkparams.tgx = [AArch64.DecodeTG0](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/AArch64-Translation?lang=en#AArch64.DecodeTG0.1)(TCR\_EL1.TG0);

walkparams.txsz = TCR\_EL1.T0SZ;

walkparams.irgn = TCR\_EL1.IRGN0;

walkparams.orgn = TCR\_EL1.ORGN0;

walkparams.sh = TCR\_EL1.SH0;

walkparams.tbi = TCR\_EL1.TBI0;

walkparams.nfd = if [HaveSVE](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/AArch64-Functions?lang=en#impl-aarch64.HaveSVE.0)() || [HaveTME](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Functions?lang=en#impl-shared.HaveTME.0)() then TCR\_EL1.NFD0 else '0';

walkparams.tbid = if [HavePACExt](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/AArch64-Functions?lang=en#impl-aarch64.HavePACExt.0)() then TCR\_EL1.TBID0 else '0';

walkparams.e0pd = if [HaveE0PDExt](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Functions?lang=en#impl-shared.HaveE0PDExt.0)() then TCR\_EL1.E0PD0 else '0';

walkparams.hpd = if [AArch64.HaveHPDExt](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Functions?lang=en#AArch64.HaveHPDExt.0)() then TCR\_EL1.HPD0 else '0';

else

walkparams.tgx = [AArch64.DecodeTG1](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/AArch64-Translation?lang=en#AArch64.DecodeTG1.1)(TCR\_EL1.TG1);

walkparams.txsz = TCR\_EL1.T1SZ;

walkparams.irgn = TCR\_EL1.IRGN1;

walkparams.orgn = TCR\_EL1.ORGN1;

walkparams.sh = TCR\_EL1.SH1;

walkparams.tbi = TCR\_EL1.TBI1;

walkparams.nfd = if [HaveSVE](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/AArch64-Functions?lang=en#impl-aarch64.HaveSVE.0)() || [HaveTME](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Functions?lang=en#impl-shared.HaveTME.0)() then TCR\_EL1.NFD1 else '0';

walkparams.tbid = if [HavePACExt](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/AArch64-Functions?lang=en#impl-aarch64.HavePACExt.0)() then TCR\_EL1.TBID1 else '0';

walkparams.e0pd = if [HaveE0PDExt](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Functions?lang=en#impl-shared.HaveE0PDExt.0)() then TCR\_EL1.E0PD1 else '0';

walkparams.hpd = if [AArch64.HaveHPDExt](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Functions?lang=en#AArch64.HaveHPDExt.0)() then TCR\_EL1.HPD1 else '0';

walkparams.mair = MAIR\_EL1;

walkparams.wxn = SCTLR\_EL1.WXN;

walkparams.ps = TCR\_EL1.IPS;

walkparams.ee = SCTLR\_EL1.EE;

walkparams.sif = SCR\_EL3.SIF;

if [EL2Enabled](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Functions?lang=en#impl-shared.EL2Enabled.0)() then

walkparams.dc = HCR\_EL2.DC;

walkparams.dct = if [HaveMTE2Ext](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Functions?lang=en#impl-shared.HaveMTE2Ext.0)() then HCR\_EL2.DCT else '0';

if [HaveTrapLoadStoreMultipleDeviceExt](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Functions?lang=en#impl-shared.HaveTrapLoadStoreMultipleDeviceExt.0)() then

walkparams.ntlsmd = SCTLR\_EL1.nTLSMD;

else

walkparams.ntlsmd = '1';

if [EL2Enabled](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Functions?lang=en#impl-shared.EL2Enabled.0)() then

if HCR\_EL2.<NV,NV1> == '01' then

case [ConstrainUnpredictable](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Functions?lang=en#impl-shared.ConstrainUnpredictable.1)([Unpredictable\_NVNV1](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Functions?lang=en#Unpredictable_NVNV1)) of

when [Constraint\_NVNV1\_00](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Functions?lang=en#Constraint_NVNV1_00) walkparams.nv1 = '0';

when [Constraint\_NVNV1\_01](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Functions?lang=en#Constraint_NVNV1_01) walkparams.nv1 = '1';

when [Constraint\_NVNV1\_11](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Functions?lang=en#Constraint_NVNV1_11) walkparams.nv1 = '1';

else

walkparams.nv1 = HCR\_EL2.NV1;

else

walkparams.nv1 = '0';

walkparams.epan = if [HavePAN3Ext](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Functions?lang=en#impl-shared.HavePAN3Ext.0)() then SCTLR\_EL1.EPAN else '0';

walkparams.cmow = if [HaveFeatCMOW](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Functions?lang=en#impl-shared.HaveFeatCMOW.0)() then SCTLR\_EL1.CMOW else '0';

walkparams.ha = if [HaveAccessFlagUpdateExt](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Functions?lang=en#impl-shared.HaveAccessFlagUpdateExt.0)() then TCR\_EL1.HA else '0';

walkparams.hd = if [HaveDirtyBitModifierExt](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Functions?lang=en#impl-shared.HaveDirtyBitModifierExt.0)() then TCR\_EL1.HD else '0';

if walkparams.tgx IN {[TGx\_4KB](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Translation?lang=en#TGx_4KB), [TGx\_16KB](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Translation?lang=en#TGx_16KB)} && [Have52BitIPAAndPASpaceExt](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Functions?lang=en#impl-shared.Have52BitIPAAndPASpaceExt.0)() then

walkparams.ds = TCR\_EL1.DS;

else

walkparams.ds = '0';

return walkparams;



### **AArch64.S1Translate**

****// AArch64.S1Translate()

// =====================

// Translate VA to IPA/PA depending on the regime

(FaultRecord, AddressDescriptor) AArch64.S1Translate([FaultRecord](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Functions?lang=en#FaultRecord) fault\_in, [Regime](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Translation?lang=en#Regime) regime,

[SecurityState](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Functions?lang=en#SecurityState) ss, bits(64) va,

[AccType](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Functions?lang=en#AccType) acctype, boolean aligned\_in,

boolean iswrite\_in, boolean ispriv)

[FaultRecord](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Functions?lang=en#FaultRecord) fault = fault\_in;

boolean aligned = aligned\_in;

boolean iswrite = iswrite\_in;

// Prepare fault fields in case a fault is detected

fault.secondstage = FALSE;

fault.s2fs1walk = FALSE;

if ![AArch64.S1Enabled](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/AArch64-Translation?lang=en#AArch64.S1Enabled.1)(regime) then

return [AArch64.S1DisabledOutput](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/AArch64-Translation?lang=en#AArch64.S1DisabledOutput.6)(fault, regime, ss, va, acctype, aligned);

walkparams = [AArch64.GetS1TTWParams](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/AArch64-Translation?lang=en#AArch64.GetS1TTWParams.2)(regime, va);

if ([AArch64.S1InvalidTxSZ](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/AArch64-Translation?lang=en#AArch64.S1InvalidTxSZ.1)(walkparams) ||

(!ispriv && walkparams.e0pd == '1') ||

(!ispriv && walkparams.nfd == '1' && [IsDataAccess](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Functions?lang=en#impl-shared.IsDataAccess.1)(acctype) && TSTATE.depth > 0) ||

(!ispriv && walkparams.nfd == '1' && acctype == [AccType\_NONFAULT](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Functions?lang=en#AccType_NONFAULT)) ||

[AArch64.VAIsOutOfRange](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/AArch64-Translation?lang=en#AArch64.VAIsOutOfRange.4)(va, acctype, regime, walkparams)) then

fault.statuscode = [Fault\_Translation](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Functions?lang=en#Fault_Translation);

fault.level = 0;

return (fault, [AddressDescriptor](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Translation?lang=en#AddressDescriptor) UNKNOWN);

[AddressDescriptor](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Translation?lang=en#AddressDescriptor) descaddress;

[TTWState](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Translation?lang=en#TTWState) walkstate;

bits(64) descriptor;

bits(64) new\_desc;

bits(64) mem\_desc;

repeat

(fault, descaddress, walkstate, descriptor) = [AArch64.S1Walk](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/AArch64-Translation?lang=en#AArch64.S1Walk.8)(fault, walkparams, va, regime,

ss, acctype, iswrite, ispriv);

if fault.statuscode != [Fault\_None](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Functions?lang=en#Fault_None) then

return (fault, [AddressDescriptor](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Translation?lang=en#AddressDescriptor) UNKNOWN);

if acctype == [AccType\_IFETCH](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Functions?lang=en#AccType_IFETCH) then

// Flag the fetched instruction is from a guarded page

[SetInGuardedPage](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/AArch64-Functions?lang=en#impl-aarch64.SetInGuardedPage.1)(walkstate.guardedpage == '1');

if [AArch64.S1HasAlignmentFault](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/AArch64-Translation?lang=en#AArch64.S1HasAlignmentFault.4)(acctype, aligned, walkparams.ntlsmd,

walkstate.memattrs) then

fault.statuscode = [Fault\_Alignment](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Functions?lang=en#Fault_Alignment);

elsif [IsAtomicRW](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Translation?lang=en#impl-shared.IsAtomicRW.1)(acctype) then

if [AArch64.S1HasPermissionsFault](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/AArch64-Translation?lang=en#AArch64.S1HasPermissionsFault.7)(regime, ss, walkstate, walkparams,

ispriv, acctype, FALSE) then

// The permission fault was not caused by lack of write permissions

fault.statuscode = [Fault\_Permission](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Functions?lang=en#Fault_Permission);

fault.write = FALSE;

elsif [AArch64.S1HasPermissionsFault](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/AArch64-Translation?lang=en#AArch64.S1HasPermissionsFault.7)(regime, ss, walkstate, walkparams,

ispriv, acctype, TRUE) then

// The permission fault \_was\_ caused by lack of write permissions

fault.statuscode = [Fault\_Permission](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Functions?lang=en#Fault_Permission);

fault.write = TRUE;

elsif [AArch64.S1HasPermissionsFault](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/AArch64-Translation?lang=en#AArch64.S1HasPermissionsFault.7)(regime, ss, walkstate, walkparams,

ispriv, acctype, iswrite) then

fault.statuscode = [Fault\_Permission](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Functions?lang=en#Fault_Permission);

new\_desc = descriptor;

if walkparams.ha == '1' && [AArch64.FaultAllowsSetAccessFlag](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/AArch64-Translation?lang=en#AArch64.FaultAllowsSetAccessFlag.1)(fault) then

// Set descriptor AF bit

new\_desc<10> = '1';

// If HW update of dirty bit is enabled, the walk state permissions

// will already reflect a configuration permitting writes.

// The update of the descriptor occurs only if the descriptor bits in

// memory do not reflect that and the access instigates a write.

if (fault.statuscode == [Fault\_None](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Functions?lang=en#Fault_None) &&

walkparams.ha == '1' &&

walkparams.hd == '1' &&

descriptor<51> == '1' && // Descriptor DBM bit

([IsAtomicRW](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Translation?lang=en#impl-shared.IsAtomicRW.1)(acctype) || iswrite) &&

!(acctype IN {[AccType\_AT](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Functions?lang=en#AccType_AT), [AccType\_ATPAN](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Functions?lang=en#AccType_ATPAN), [AccType\_IC](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Functions?lang=en#AccType_IC), [AccType\_DC](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Functions?lang=en#AccType_DC)})) then

// Clear descriptor AP[2] bit permitting stage 1 writes

new\_desc<7> = '0';

[AddressDescriptor](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Translation?lang=en#AddressDescriptor) descupdateaddress;

[FaultRecord](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Functions?lang=en#FaultRecord) s2fault;

// Either the access flag was clear or AP<2> is set

if new\_desc != descriptor then

if regime == [Regime\_EL10](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Translation?lang=en#Regime_EL10) && [EL2Enabled](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Functions?lang=en#impl-shared.EL2Enabled.0)() then

s1aarch64 = TRUE;

s2fs1walk = TRUE;

aligned = TRUE;

iswrite = TRUE;

(s2fault, descupdateaddress) = [AArch64.S2Translate](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/AArch64-Translation?lang=en#AArch64.S2Translate.9)(fault, descaddress, s1aarch64,

ss, s2fs1walk, [AccType\_ATOMICRW](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Functions?lang=en#AccType_ATOMICRW),

aligned, iswrite, ispriv);

if s2fault.statuscode != [Fault\_None](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Functions?lang=en#Fault_None) then

return (s2fault, [AddressDescriptor](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Translation?lang=en#AddressDescriptor) UNKNOWN);

else

descupdateaddress = descaddress;

(fault, mem\_desc) = [AArch64.MemSwapTableDesc](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/AArch64-Translation?lang=en#AArch64.MemSwapTableDesc.5)(fault, descriptor, new\_desc,

walkparams.ee, descupdateaddress);

until new\_desc == descriptor || mem\_desc == new\_desc;

if fault.statuscode != [Fault\_None](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Functions?lang=en#Fault_None) then

return (fault, [AddressDescriptor](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Translation?lang=en#AddressDescriptor) UNKNOWN);

// Output Address

oa = [StageOA](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Translation?lang=en#impl-shared.StageOA.3)(va, walkparams.tgx, walkstate);

[MemoryAttributes](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Functions?lang=en#MemoryAttributes) memattrs;

if (acctype == [AccType\_IFETCH](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Functions?lang=en#AccType_IFETCH) &&

(walkstate.memattrs.memtype == [MemType\_Device](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Functions?lang=en#MemType_Device) || ![AArch64.S1ICacheEnabled](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/AArch64-Translation?lang=en#AArch64.S1ICacheEnabled.1)(regime))) then

// Treat memory attributes as Normal Non-Cacheable

memattrs = [NormalNCMemAttr](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Translation?lang=en#impl-shared.NormalNCMemAttr.0)();

memattrs.xs = walkstate.memattrs.xs;

elsif (acctype != [AccType\_IFETCH](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Functions?lang=en#AccType_IFETCH) && ![AArch64.S1DCacheEnabled](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/AArch64-Translation?lang=en#AArch64.S1DCacheEnabled.1)(regime) &&

walkstate.memattrs.memtype == [MemType\_Normal](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Functions?lang=en#MemType_Normal)) then

// Treat memory attributes as Normal Non-Cacheable

memattrs = [NormalNCMemAttr](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Translation?lang=en#impl-shared.NormalNCMemAttr.0)();

memattrs.xs = walkstate.memattrs.xs;

// The effect of SCTLR\_ELx.C when '0' is Constrained UNPREDICTABLE

// on the Tagged attribute

if [HaveMTE2Ext](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Functions?lang=en#impl-shared.HaveMTE2Ext.0)() && walkstate.memattrs.tagged then

memattrs.tagged = [ConstrainUnpredictableBool](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Functions?lang=en#impl-shared.ConstrainUnpredictableBool.1)([Unpredictable\_S1CTAGGED](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Functions?lang=en#Unpredictable_S1CTAGGED));

else

memattrs = walkstate.memattrs;

// Shareability value of stage 1 translation subject to stage 2 is IMPLEMENTATION DEFINED

// to be either effective value or descriptor value

if (regime == [Regime\_EL10](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Translation?lang=en#Regime_EL10) && [EL2Enabled](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Functions?lang=en#impl-shared.EL2Enabled.0)() && HCR\_EL2.VM == '1' &&

!(boolean IMPLEMENTATION\_DEFINED "Apply effective shareability at stage 1")) then

memattrs.shareability = walkstate.memattrs.shareability;

else

memattrs.shareability = [EffectiveShareability](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Translation?lang=en#impl-shared.EffectiveShareability.1)(memattrs);

if acctype == [AccType\_ATOMICLS64](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Functions?lang=en#AccType_ATOMICLS64) && memattrs.memtype == [MemType\_Normal](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Functions?lang=en#MemType_Normal) then

if memattrs.inner.attrs != [MemAttr\_NC](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Functions?lang=en#MemAttr_NC) || memattrs.outer.attrs != [MemAttr\_NC](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Functions?lang=en#MemAttr_NC) then

fault.statuscode = [Fault\_Exclusive](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Functions?lang=en#Fault_Exclusive);

return (fault, [AddressDescriptor](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Translation?lang=en#AddressDescriptor) UNKNOWN);

ipa = [CreateAddressDescriptor](https://developer.arm.com/documentation/ddi0602/2021-12/Shared-Pseudocode/Shared-Translation?lang=en#impl-shared.CreateAddressDescriptor.3)(va, oa, memattrs);

return (fault, ipa);

