

HIP API

Generated by Doxygen 1.8.20

1 Module Index	1
1.1 Modules	1
2 Hierarchical Index	3
2.1 Class Hierarchy	3
3 Class Index	7
3.1 Class List	7
4 Module Documentation	11
4.1 HIP API	11
4.1.1 Detailed Description	12
4.1.2 Function Documentation	12
4.1.2.1 hipRegisterApiCallback()	12
4.2 Initialization and Version	13
4.2.1 Detailed Description	13
4.2.2 Function Documentation	13
4.2.2.1 hipDeviceComputeCapability()	13
4.2.2.2 hipDeviceGet()	14
4.2.2.3 hipDeviceGetByPCIBusId()	14
4.2.2.4 hipDeviceGetName()	14
4.2.2.5 hipDeviceGetP2PAttribute()	16
4.2.2.6 hipDeviceGetPCIBusId()	16
4.2.2.7 hipDeviceTotalMem()	17
4.2.2.8 hipDriverGetVersion()	17
4.2.2.9 hipInit()	18
4.2.2.10 hipRuntimeGetVersion()	18
4.3 Device Management	19
4.3.1 Detailed Description	19
4.3.2 Function Documentation	20
4.3.2.1 hipChooseDevice()	20
4.3.2.2 hipDeviceGetAttribute()	20
4.3.2.3 hipDeviceGetCacheConfig()	20
4.3.2.4 hipDeviceGetLimit()	22
4.3.2.5 hipDeviceGetSharedMemConfig()	22
4.3.2.6 hipDeviceReset()	23
4.3.2.7 hipDeviceSetCacheConfig()	23
4.3.2.8 hipDeviceSetSharedMemConfig()	23
4.3.2.9 hipDeviceSynchronize()	24
4.3.2.10 hipExtGetLinkTypeAndHopCount()	24
4.3.2.11 hipGetDevice()	25
4.3.2.12 hipGetDeviceCount()	25
4.3.2.13 hipGetDeviceFlags()	25

4.3.2.14	hipGetDeviceProperties()	26
4.3.2.15	hipIpcCloseMemHandle()	26
4.3.2.16	hipIpcGetMemHandle()	28
4.3.2.17	hipIpcOpenMemHandle()	28
4.3.2.18	hipSetDevice()	29
4.3.2.19	hipSetDeviceFlags()	30
4.4	Execution Control	31
4.4.1	Detailed Description	31
4.4.2	Function Documentation	31
4.4.2.1	hipFuncSetAttribute()	31
4.4.2.2	hipFuncSetCacheConfig()	31
4.4.2.3	hipFuncSetSharedMemConfig()	32
4.5	Error Handling	33
4.5.1	Detailed Description	33
4.5.2	Function Documentation	33
4.5.2.1	hipGetErrorName()	33
4.5.2.2	hipGetErrorString()	33
4.5.2.3	hipGetLastError()	34
4.5.2.4	hipPeekAtLastError()	34
4.6	Stream Management	35
4.6.1	Detailed Description	36
4.6.2	Typedef Documentation	36
4.6.2.1	hipStreamCallback_t [1/2]	36
4.6.2.2	hipStreamCallback_t [2/2]	36
4.6.3	Function Documentation	36
4.6.3.1	hipDeviceGetStreamPriorityRange()	36
4.6.3.2	hipExtStreamCreateWithCUMask()	36
4.6.3.3	hipExtStreamGetCUMask()	37
4.6.3.4	hipStreamAddCallback()	37
4.6.3.5	hipStreamCreate()	38
4.6.3.6	hipStreamCreateWithFlags()	38
4.6.3.7	hipStreamCreateWithPriority()	39
4.6.3.8	hipStreamDestroy()	39
4.6.3.9	hipStreamGetFlags()	40
4.6.3.10	hipStreamGetPriority()	40
4.6.3.11	hipStreamQuery()	41
4.6.3.12	hipStreamSynchronize()	41
4.6.3.13	hipStreamWaitEvent()	41
4.6.3.14	hipStreamWaitValue32()	42
4.6.3.15	hipStreamWaitValue64()	43
4.6.3.16	hipStreamWriteValue32()	43
4.6.3.17	hipStreamWriteValue64()	44

4.7 Event Management	45
4.7.1 Detailed Description	45
4.7.2 Function Documentation	45
4.7.2.1 hipEventCreate()	45
4.7.2.2 hipEventCreateWithFlags()	45
4.7.2.3 hipEventDestroy()	46
4.7.2.4 hipEventElapsedTime()	46
4.7.2.5 hipEventQuery()	47
4.7.2.6 hipEventRecord()	47
4.7.2.7 hipEventSynchronize()	48
4.8 Memory Management	49
4.8.1 Detailed Description	52
4.8.2 Function Documentation	52
4.8.2.1 hipDestroyExternalMemory()	52
4.8.2.2 hipDrvMemcpy3D()	52
4.8.2.3 hipDrvMemcpy3DAsync()	53
4.8.2.4 hipExternalMemoryGetMappedBuffer()	53
4.8.2.5 hipExtMallocWithFlags()	54
4.8.2.6 hipFree()	54
4.8.2.7 hipFreeArray()	54
4.8.2.8 hipFreeHost()	55
4.8.2.9 hipFreeMipmappedArray()	55
4.8.2.10 hipGetMipmappedArrayLevel()	55
4.8.2.11 hipHostAlloc()	56
4.8.2.12 hipHostFree()	56
4.8.2.13 hipHostGetDevicePointer()	57
4.8.2.14 hipHostGetFlags()	57
4.8.2.15 hipHostMalloc()	57
4.8.2.16 hipHostRegister()	58
4.8.2.17 hipHostUnregister()	59
4.8.2.18 hipImportExternalMemory()	59
4.8.2.19 hipMalloc()	59
4.8.2.20 hipMalloc3DArray()	60
4.8.2.21 hipMallocArray()	60
4.8.2.22 hipMallocHost()	61
4.8.2.23 hipMallocMipmappedArray()	61
4.8.2.24 hipMallocPitch()	62
4.8.2.25 hipMemAllocHost()	62
4.8.2.26 hipMemAllocPitch()	63
4.8.2.27 hipMemcpy()	63
4.8.2.28 hipMemcpy2D()	64
4.8.2.29 hipMemcpy2DAsync()	65

4.8.2.30	hipMemcpy2DFromArray()	65
4.8.2.31	hipMemcpy2DFromArrayAsync()	67
4.8.2.32	hipMemcpy2DToArray()	68
4.8.2.33	hipMemcpy2DToArrayAsync()	68
4.8.2.34	hipMemcpy3D()	69
4.8.2.35	hipMemcpy3DAsync()	69
4.8.2.36	hipMemcpyAsync()	70
4.8.2.37	hipMemcpyAtoH()	70
4.8.2.38	hipMemcpyDtoD()	71
4.8.2.39	hipMemcpyDtoDAsync()	71
4.8.2.40	hipMemcpyDtoH()	73
4.8.2.41	hipMemcpyDtoHAsync()	73
4.8.2.42	hipMemcpyFromArray()	74
4.8.2.43	hipMemcpyHtoA()	74
4.8.2.44	hipMemcpyHtoD()	75
4.8.2.45	hipMemcpyHtoDAsync()	75
4.8.2.46	hipMemcpyParam2D()	77
4.8.2.47	hipMemcpyParam2DAsync()	77
4.8.2.48	hipMemcpyToArray()	78
4.8.2.49	hipMemGetInfo()	78
4.8.2.50	hipMemset()	79
4.8.2.51	hipMemset2D()	79
4.8.2.52	hipMemset2DAsync()	79
4.8.2.53	hipMemset3D()	80
4.8.2.54	hipMemset3DAsync()	80
4.8.2.55	hipMemsetAsync()	81
4.8.2.56	hipMemsetD16()	81
4.8.2.57	hipMemsetD16Async()	81
4.8.2.58	hipMemsetD32()	82
4.8.2.59	hipMemsetD32Async()	82
4.8.2.60	hipMemsetD8()	83
4.8.2.61	hipMemsetD8Async()	83
4.8.2.62	hipPointerGetAttributes()	83
4.9	PeerToPeer Device Memory Access	85
4.9.1	Detailed Description	85
4.9.2	Function Documentation	85
4.9.2.1	hipDeviceCanAccessPeer()	85
4.9.2.2	hipDeviceDisablePeerAccess()	86
4.9.2.3	hipDeviceEnablePeerAccess()	86
4.9.2.4	hipMemcpyPeer()	86
4.9.2.5	hipMemcpyPeerAsync()	87
4.9.2.6	hipMemGetAddressRange()	87

4.10 Context Management	89
4.10.1 Detailed Description	89
4.10.2 Function Documentation	89
4.10.2.1 hipDevicePrimaryCtxGetState()	89
4.10.2.2 hipDevicePrimaryCtxRelease()	89
4.10.2.3 hipDevicePrimaryCtxReset()	90
4.10.2.4 hipDevicePrimaryCtxRetain()	90
4.10.2.5 hipDevicePrimaryCtxSetFlags()	91
4.11 Module Management	92
4.11.1 Detailed Description	92
4.11.2 Function Documentation	93
4.11.2.1 hipExtLaunchMultiKernelMultiDevice()	93
4.11.2.2 hipExtModuleLaunchKernel()	93
4.11.2.3 hipFuncGetAttribute()	94
4.11.2.4 hipFuncGetAttributes()	94
4.11.2.5 hipLaunchCooperativeKernel()	95
4.11.2.6 hipLaunchCooperativeKernelMultiDevice()	95
4.11.2.7 hipModuleGetFunction()	96
4.11.2.8 hipModuleGetTexRef()	96
4.11.2.9 hipModuleLaunchKernel()	96
4.11.2.10 hipModuleLoad()	97
4.11.2.11 hipModuleLoadData()	97
4.11.2.12 hipModuleLoadDataEx()	98
4.11.2.13 hipModuleUnload()	98
4.12 Occupancy	99
4.12.1 Detailed Description	99
4.12.2 Function Documentation	99
4.12.2.1 hipModuleOccupancyMaxActiveBlocksPerMultiprocessor()	99
4.12.2.2 hipModuleOccupancyMaxActiveBlocksPerMultiprocessorWithFlags()	99
4.12.2.3 hipModuleOccupancyMaxPotentialBlockSize()	100
4.12.2.4 hipModuleOccupancyMaxPotentialBlockSizeWithFlags()	100
4.12.2.5 hipOccupancyMaxActiveBlocksPerMultiprocessor()	101
4.12.2.6 hipOccupancyMaxActiveBlocksPerMultiprocessorWithFlags()	101
4.12.2.7 hipOccupancyMaxPotentialBlockSize()	101
4.13 Profiler Control[Deprecated]	103
4.13.1 Detailed Description	103
4.13.2 Function Documentation	103
4.13.2.1 hipProfilerStart()	103
4.13.2.2 hipProfilerStop()	103
4.14 Launch API to support the triple-chevron syntax	104
4.14.1 Detailed Description	104
4.14.2 Function Documentation	104

4.14.2.1	__hipPopCallConfiguration()	104
4.14.2.2	__hipPushCallConfiguration()	105
4.14.2.3	hipConfigureCall()	105
4.14.2.4	hipDrvMemcpy2DUnaligned()	105
4.14.2.5	hipLaunchByPtr()	107
4.14.2.6	hipLaunchKernel()	107
4.14.2.7	hipSetupArgument()	107
4.15	Texture Management	109
4.15.1	Detailed Description	110
4.16	Global enum and defines	111
4.16.1	Detailed Description	117
4.16.2	Macro Definition Documentation	117
4.16.2.1	hipDeviceScheduleSpin [1/2]	117
4.16.2.2	hipDeviceScheduleSpin [2/2]	117
4.16.2.3	hipDeviceScheduleYield [1/2]	118
4.16.2.4	hipDeviceScheduleYield [2/2]	118
4.16.2.5	hipEventDefault [1/2]	118
4.16.2.6	hipEventDefault [2/2]	118
4.16.2.7	hipEventInterprocess [1/2]	118
4.16.2.8	hipEventInterprocess [2/2]	118
4.16.2.9	hipEventReleaseToSystem [1/2]	118
4.16.2.10	hipEventReleaseToSystem [2/2]	118
4.16.2.11	hipHostMallocCoherent [1/2]	118
4.16.2.12	hipHostMallocCoherent [2/2]	119
4.16.2.13	hipHostMallocDefault [1/2]	119
4.16.2.14	hipHostMallocDefault [2/2]	119
4.16.2.15	hipHostMallocMapped [1/2]	119
4.16.2.16	hipHostMallocMapped [2/2]	119
4.16.2.17	hipHostMallocNonCoherent [1/2]	119
4.16.2.18	hipHostMallocNonCoherent [2/2]	119
4.16.2.19	hipHostRegisterDefault [1/2]	119
4.16.2.20	hipHostRegisterDefault [2/2]	119
4.16.2.21	hipHostRegisterMapped [1/2]	119
4.16.2.22	hipHostRegisterMapped [2/2]	120
4.16.2.23	hipMemAttachSingle [1/2]	120
4.16.2.24	hipMemAttachSingle [2/2]	120
4.16.2.25	hipStreamDefault [1/2]	120
4.16.2.26	hipStreamDefault [2/2]	120
4.16.3	Typedef Documentation	120
4.16.3.1	dim3 [1/2]	120
4.16.3.2	dim3 [2/2]	120
4.16.3.3	hipFuncAttribute [1/2]	120

4.16.3.4 hipFuncAttribute [2/2]	120
4.16.3.5 hipFuncCache_t [1/2]	121
4.16.3.6 hipFuncCache_t [2/2]	121
4.16.3.7 hipSharedMemConfig [1/2]	121
4.16.3.8 hipSharedMemConfig [2/2]	121
4.16.4 Enumeration Type Documentation	121
4.16.4.1 hipDeviceAttribute_t	121
4.16.4.2 hipFuncAttribute [1/2]	124
4.16.4.3 hipFuncAttribute [2/2]	124
4.16.4.4 hipFuncCache_t [1/2]	125
4.16.4.5 hipFuncCache_t [2/2]	125
4.16.4.6 hipMemoryAdvise [1/2]	125
4.16.4.7 hipMemoryAdvise [2/2]	126
4.16.4.8 hipMemRangeAttribute [1/2]	126
4.16.4.9 hipMemRangeAttribute [2/2]	127
4.16.4.10 hipSharedMemConfig [1/2]	127
4.16.4.11 hipSharedMemConfig [2/2]	128
4.17 Managed Memory (ROCm HMM)	129
4.17.1 Detailed Description	129
4.17.2 Function Documentation	129
4.17.2.1 hipMallocManaged()	129
4.17.2.2 hipMemAdvise()	129
4.17.2.3 hipMemPrefetchAsync()	130
4.17.2.4 hipMemRangeGetAttribute()	130
4.17.2.5 hipMemRangeGetAttributes()	131
4.17.2.6 hipStreamAttachMemAsync()	131
4.18 Context Management [Deprecated]	132
4.18.1 Detailed Description	132
4.18.2 Function Documentation	132
4.18.2.1 hipCtxCreate()	132
4.18.2.2 hipCtxDestroy()	133
4.18.2.3 hipCtxDisablePeerAccess()	133
4.18.2.4 hipCtxEnablePeerAccess()	134
4.18.2.5 hipCtxGetApiVersion()	134
4.18.2.6 hipCtxGetCacheConfig()	135
4.18.2.7 hipCtxGetCurrent()	135
4.18.2.8 hipCtxGetDevice()	136
4.18.2.9 hipCtxGetFlags()	136
4.18.2.10 hipCtxGetSharedMemConfig()	136
4.18.2.11 hipCtxPopCurrent()	137
4.18.2.12 hipCtxPushCurrent()	137
4.18.2.13 hipCtxSetCacheConfig()	137

4.18.2.14 hipCtxSetCurrent()	138
4.18.2.15 hipCtxSetSharedMemConfig()	138
4.18.2.16 hipCtxSynchronize()	139
4.19 Texture Management [Deprecated]	140
4.19.1 Detailed Description	140
5 Class Documentation	141
5.1 __half2_raw Struct Reference	141
5.2 __half_raw Struct Reference	141
5.3 __hip_enable_if< __B, __T > Struct Template Reference	141
5.4 __hip_enable_if< true, __T > Struct Template Reference	141
5.5 char1 Union Reference	141
5.6 char16 Union Reference	141
5.7 char2 Union Reference	141
5.8 char3 Union Reference	142
5.9 char4 Union Reference	142
5.10 char8 Union Reference	142
5.11 dim3 Struct Reference	142
5.11.1 Detailed Description	142
5.12 double1 Union Reference	142
5.13 double16 Union Reference	142
5.14 double2 Union Reference	142
5.15 double3 Union Reference	143
5.16 double4 Union Reference	143
5.17 double8 Union Reference	143
5.18 float1 Union Reference	143
5.19 float16 Union Reference	143
5.20 float2 Union Reference	143
5.21 float3 Union Reference	143
5.22 float4 Union Reference	143
5.23 float8 Union Reference	143
5.24 gl_dim3 Struct Reference	144
5.25 grid_launch_parm Struct Reference	144
5.25.1 Member Data Documentation	144
5.25.1.1 av	144
5.25.1.2 barrier_bit	144
5.25.1.3 cf	144
5.25.1.4 dynamic_group_mem_bytes	145
5.25.1.5 launch_fence	145
5.26 grid_launch_parm_cxx Class Reference	145
5.27 HIP_ARRAY3D_DESCRIPTOR Struct Reference	145
5.28 HIP_ARRAY_DESCRIPTOR Struct Reference	145

5.29 hip_bfloat16 Struct Reference	145
5.29.1 Detailed Description	146
5.30 hip_Memcpy2D Struct Reference	146
5.31 HIP_MEMCPY3D Struct Reference	146
5.32 HIP_RESOURCE_DESC_st Struct Reference	147
5.32.1 Member Data Documentation	147
5.32.1.1 devPtr	147
5.32.1.2 flags	147
5.32.1.3 format	147
5.32.1.4 hArray	147
5.32.1.5 height	148
5.32.1.6 hMipmappedArray	148
5.32.1.7 numChannels	148
5.32.1.8 pitchInBytes	148
5.32.1.9 resType	148
5.32.1.10 sizeInBytes	148
5.32.1.11 width	148
5.33 HIP_RESOURCE_VIEW_DESC_st Struct Reference	148
5.33.1 Detailed Description	148
5.33.2 Member Data Documentation	148
5.33.2.1 depth	149
5.33.2.2 firstLayer	149
5.33.2.3 firstMipmapLevel	149
5.33.2.4 format	149
5.33.2.5 height	149
5.33.2.6 lastLayer	149
5.33.2.7 lastMipmapLevel	149
5.33.2.8 width	149
5.34 HIP_TEXTURE_DESC_st Struct Reference	149
5.34.1 Detailed Description	149
5.34.2 Member Data Documentation	150
5.34.2.1 addressMode	150
5.34.2.2 borderColor	150
5.34.2.3 filterMode	150
5.34.2.4 flags	150
5.34.2.5 maxAnisotropy	150
5.34.2.6 maxMipmapLevelClamp	150
5.34.2.7 minMipmapLevelClamp	150
5.34.2.8 mipmapFilterMode	150
5.34.2.9 mipmapLevelBias	150
5.35 hipArray Struct Reference	150
5.36 hipChannelFormatDesc Struct Reference	151

5.37 hipDeviceArch_t Struct Reference	151
5.38 hipDeviceProp_t Struct Reference	152
5.38.1 Detailed Description	154
5.38.2 Member Data Documentation	154
5.38.2.1 clockInstructionRate	154
5.38.2.2 cooperativeMultiDeviceUnmatchedBlockDim	154
5.38.2.3 cooperativeMultiDeviceUnmatchedFunc	154
5.38.2.4 cooperativeMultiDeviceUnmatchedGridDim	154
5.38.2.5 cooperativeMultiDeviceUnmatchedSharedMem	154
5.38.2.6 major	154
5.38.2.7 minor	154
5.38.2.8 pageableMemoryAccess	154
5.39 hipExtent Struct Reference	154
5.40 hipExternalMemoryBufferDesc_st Struct Reference	155
5.41 hipExternalMemoryHandleDesc_st Struct Reference	155
5.42 hipFuncAttributes Struct Reference	155
5.43 hipIpcEventHandle_st Struct Reference	155
5.44 hipIpcMemHandle_st Struct Reference	155
5.45 hipLaunchParams_t Struct Reference	156
5.46 hipMemcpy3DParms Struct Reference	156
5.47 hipMipmappedArray Struct Reference	156
5.48 hipPitchedPtr Struct Reference	156
5.49 hipPointerAttribute_t Struct Reference	157
5.49.1 Detailed Description	157
5.50 hipPos Struct Reference	157
5.51 hipResourceDesc Struct Reference	157
5.51.1 Detailed Description	157
5.52 hipResourceViewDesc Struct Reference	158
5.52.1 Detailed Description	158
5.53 hipTextureDesc Struct Reference	158
5.53.1 Detailed Description	158
5.54 int1 Union Reference	158
5.55 int16 Union Reference	158
5.56 int2 Union Reference	158
5.57 int3 Union Reference	159
5.58 int4 Union Reference	159
5.59 int8 Union Reference	159
5.60 hip_impl::is_callable< Call > Struct Template Reference	159
5.61 hip_impl::is_callable_impl< typename, typename > Struct Template Reference	159
5.62 hip_impl::is_callable_impl< F(Ts...), void_t< decltype(simple_invoke(std::declval< F >()), std::declval< Ts >()...) > > Struct Template Reference	159
5.63 hip_impl::kernarg Class Reference	160

5.64 hip_impl::kernargs_size_align Class Reference	160
5.65 long1 Union Reference	160
5.66 long16 Union Reference	160
5.67 long2 Union Reference	160
5.68 long3 Union Reference	160
5.69 long4 Union Reference	161
5.70 long8 Union Reference	161
5.71 longlong1 Union Reference	161
5.72 longlong16 Union Reference	161
5.73 longlong2 Union Reference	161
5.74 longlong3 Union Reference	161
5.75 longlong4 Union Reference	161
5.76 longlong8 Union Reference	161
5.77 hip_impl::program_state Class Reference	161
5.78 short1 Union Reference	162
5.79 short16 Union Reference	162
5.80 short2 Union Reference	162
5.81 short3 Union Reference	162
5.82 short4 Union Reference	162
5.83 short8 Union Reference	162
5.84 surfaceReference Struct Reference	163
5.84.1 Detailed Description	163
5.85 TData Union Reference	163
5.86 textureReference Struct Reference	163
5.86.1 Detailed Description	163
5.87 uchar1 Union Reference	163
5.88 uchar16 Union Reference	163
5.89 uchar2 Union Reference	164
5.90 uchar2Holder Struct Reference	164
5.91 uchar3 Union Reference	164
5.92 uchar4 Union Reference	164
5.93 uchar8 Union Reference	164
5.94 ucharHolder Struct Reference	164
5.95 uint1 Union Reference	164
5.96 uint16 Union Reference	165
5.97 uint2 Union Reference	165
5.98 uint3 Union Reference	165
5.99 uint4 Union Reference	165
5.100 uint8 Union Reference	165
5.101 ulong1 Union Reference	165
5.102 ulong16 Union Reference	165
5.103 ulong2 Union Reference	165

5.104 ulong3 Union Reference	165
5.105 ulong4 Union Reference	166
5.106 ulong8 Union Reference	166
5.107 ulonglong1 Union Reference	166
5.108 ulonglong16 Union Reference	166
5.109 ulonglong2 Union Reference	166
5.110 ulonglong3 Union Reference	166
5.111 ulonglong4 Union Reference	166
5.112 ulonglong8 Union Reference	166
5.113 ushort1 Union Reference	166
5.114 ushort16 Union Reference	167
5.115 ushort2 Union Reference	167
5.116 ushort3 Union Reference	167
5.117 ushort4 Union Reference	167
5.118 ushort8 Union Reference	167

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

HIP API	11
Initialization and Version	13
Device Management	19
Execution Control	31
Error Handling	33
Stream Management	35
Event Management	45
Memory Management	49
Managed Memory (ROCm HMM)	129
PeerToPeer Device Memory Access	85
Context Management	89
Context Management [Deprecated]	132
Module Management	92
Occupancy	99
Profiler Control[Deprecated]	103
Launch API to support the triple-chevron syntax	104
Texture Management	109
Texture Management [Deprecated]	140
Global enum and defines	111

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

__half2_raw	141
__half_raw	141
__hip_enable_if< __B, __T >	141
__hip_enable_if< true, __T >	141
char1	141
char16	141
char2	141
char3	142
char4	142
char8	142
dim3	142
double1	142
double16	142
double2	142
double3	143
double4	143
double8	143
false_type	
hip_impl::is_callable_impl< typename, typename >	159
hip_impl::is_callable_impl< Call >	159
hip_impl::is_callable< Call >	159
float1	143
float16	143
float2	143
float3	143
float4	143
float8	143
gl_dim3	144
grid_launch_parm	144
grid_launch_parm_cxx	145
HIP_ARRAY3D_DESCRIPTOR	145
HIP_ARRAY_DESCRIPTOR	145
hip_bfloat16	145
hip_Memcpy2D	146
HIP_MEMCPY3D	146

HIP_RESOURCE_DESC_st	147
HIP_RESOURCE_VIEW_DESC_st	148
HIP_TEXTURE_DESC_st	149
hipArray	150
hipChannelFormatDesc	151
hipDeviceArch_t	151
hipDeviceProp_t	152
hipExtent	154
hipExternalMemoryBufferDesc_st	155
hipExternalMemoryHandleDesc_st	155
hipFuncAttributes	155
hipIpcEventHandle_st	155
hipIpcMemHandle_st	155
hipLaunchParams_t	156
hipMemcpy3DParms	156
hipMipmappedArray	156
hipPitchedPtr	156
hipPointerAttribute_t	157
hipPos	157
hipResourceDesc	157
hipResourceViewDesc	158
hipTextureDesc	158
int1	158
int16	158
int2	158
int3	159
int4	159
int8	159
hip_impl::kernarg	160
hip_impl::kernargs_size_align	160
long1	160
long16	160
long2	160
long3	160
long4	161
long8	161
longlong1	161
longlong16	161
longlong2	161
longlong3	161
longlong4	161
longlong8	161
hip_impl::program_state	161
short1	162
short16	162
short2	162
short3	162
short4	162
short8	162
surfaceReference	163
TData	163
textureReference	163
true_type	
hip_impl::is_callable_impl< F(Ts...), void_t< decltype(simple_invoke(std::declval< F >()), std::declval< Ts >())>>	159
uchar1	163
uchar16	163
uchar2	164

uchar2Holder	164
uchar3	164
uchar4	164
uchar8	164
ucharHolder	164
uint1	164
uint16	165
uint2	165
uint3	165
uint4	165
uint8	165
ulong1	165
ulong16	165
ulong2	165
ulong3	165
ulong4	166
ulong8	166
ulonglong1	166
ulonglong16	166
ulonglong2	166
ulonglong3	166
ulonglong4	166
ulonglong8	166
ushort1	166
ushort16	167
ushort2	167
ushort3	167
ushort4	167
ushort8	167

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

__half2_raw	141
__half_raw	141
__hip_enable_if< __B, __T >	141
__hip_enable_if< true, __T >	141
char1	141
char16	141
char2	141
char3	142
char4	142
char8	142
dim3	142
double1	142
double16	142
double2	142
double3	143
double4	143
double8	143
float1	143
float16	143
float2	143
float3	143
float4	143
float8	143
gl_dim3	144
grid_launch_parm	144
grid_launch_parm_cxx	145
HIP_ARRAY3D_DESCRIPTOR	145
HIP_ARRAY_DESCRIPTOR	145
hip_bfloat16	
Struct to represent a 16 bit brain floating point number	145
hip_Memcpy2D	146
HIP_MEMCPY3D	146
HIP_RESOURCE_DESC_st	147
HIP_RESOURCE_VIEW_DESC_st	148
HIP_TEXTURE_DESC_st	149

hipArray	150
hipChannelFormatDesc	151
hipDeviceArch_t	151
hipDeviceProp_t	152
hipExtent	154
hipExternalMemoryBufferDesc_st	155
hipExternalMemoryHandleDesc_st	155
hipFuncAttributes	155
hipIpcEventHandle_st	155
hipIpcMemHandle_st	155
hipLaunchParams_t	156
hipMemcpy3DParms	156
hipMipmappedArray	156
hipPitchedPtr	156
hipPointerAttribute_t	157
hipPos	157
hipResourceDesc	157
hipResourceViewDesc	158
hipTextureDesc	158
int1	158
int16	158
int2	158
int3	159
int4	159
int8	159
hip_impl::is_callable< Call >	159
hip_impl::is_callable_impl< typename, typename >	159
hip_impl::is_callable_impl< F(Ts...), void_t< decltype(simple_invoke(std::declval< F >()), std::declval< Ts >()...)) >	159
hip_impl::kernarg	160
hip_impl::kernargs_size_align	160
long1	160
long16	160
long2	160
long3	160
long4	161
long8	161
longlong1	161
longlong16	161
longlong2	161
longlong3	161
longlong4	161
longlong8	161
hip_impl::program_state	161
short1	162
short16	162
short2	162
short3	162
short4	162
short8	162
surfaceReference	163
TData	163
textureReference	163
uchar1	163
uchar16	163
uchar2	164
uchar2Holder	164
uchar3	164

uchar4	164
uchar8	164
ucharHolder	164
uint1	164
uint16	165
uint2	165
uint3	165
uint4	165
uint8	165
ulong1	165
ulong16	165
ulong2	165
ulong3	165
ulong4	166
ulong8	166
ulonglong1	166
ulonglong16	166
ulonglong2	166
ulonglong3	166
ulonglong4	166
ulonglong8	166
ushort1	166
ushort16	167
ushort2	167
ushort3	167
ushort4	167
ushort8	167

Chapter 4

Module Documentation

4.1 HIP API

Modules

- [Initialization and Version](#)
- [Device Management](#)
- [Execution Control](#)
- [Error Handling](#)
- [Stream Management](#)
- [Event Management](#)
- [Memory Management](#)
- [PeerToPeer Device Memory Access](#)
- [Context Management](#)
- [Module Management](#)
- [Occupancy](#)
- [Profiler Control\[Deprecated\]](#)
- [Launch API to support the triple-chevron syntax](#)
- [Texture Management](#)

Functions

- `hipError_t hipTexRefSetBorderColor` ([textureReference](#) *texRef, float *pBorderColor)
- `hipError_t hipTexRefSetMipmapFilterMode` ([textureReference](#) *texRef, enum hipTextureFilterMode fm)
- `hipError_t hipTexRefSetMipmapLevelBias` ([textureReference](#) *texRef, float bias)
- `hipError_t hipTexRefSetMipmapLevelClamp` ([textureReference](#) *texRef, float minMipMapLevelClamp, float maxMipMapLevelClamp)
- `hipError_t hipTexRefSetMipmappedArray` ([textureReference](#) *texRef, struct [hipMipmappedArray](#) *mipmappedArray, unsigned int Flags)
- `hipError_t hipMipmappedArrayCreate` ([hipMipmappedArray_t](#) *pHandle, [HIP_ARRAY3D_DESCRIPTOR](#) *pMipmappedArrayDesc, unsigned int numMipmapLevels)
- `hipError_t hipMipmappedArrayDestroy` ([hipMipmappedArray_t](#) hMipmappedArray)
- `hipError_t hipMipmappedArrayGetLevel` ([hipArray_t](#) *pLevelArray, [hipMipmappedArray_t](#) hMipMappedArray, unsigned int level)
- `hipError_t hipRegisterApiCallback` (uint32_t id, void *fun, void *arg)
- `hipError_t hipRemoveApiCallback` (uint32_t id)
- `hipError_t hipRegisterActivityCallback` (uint32_t id, void *fun, void *arg)
- `hipError_t hipRemoveActivityCallback` (uint32_t id)
- `const char * hipApiName` (uint32_t id)
- `const char * hipKernelNameRef` (const hipFunction_t f)
- `const char * hipKernelNameRefByPtr` (const void *hostFunction, hipStream_t stream)
- `int hipGetStreamDeviceld` (hipStream_t stream)

4.1.1 Detailed Description

Defines the HIP API. See the individual sections for more information.

4.1.2 Function Documentation

4.1.2.1 hipRegisterApiCallback()

```
hipError_t hipRegisterApiCallback (
    uint32_t id,
    void * fun,
    void * arg )
```

Callback/Activity API

4.2 Initialization and Version

Functions

- hipError_t [hipInit](#) (unsigned int flags)
Explicitly initializes the HIP runtime.
- hipError_t [hipDriverGetVersion](#) (int *driverVersion)
Returns the approximate HIP driver version.
- hipError_t [hipRuntimeGetVersion](#) (int *runtimeVersion)
Returns the approximate HIP Runtime version.
- hipError_t [hipDeviceGet](#) (hipDevice_t *device, int ordinal)
Returns a handle to a compute device.
- hipError_t [hipDeviceComputeCapability](#) (int *major, int *minor, hipDevice_t device)
Returns the compute capability of the device.
- hipError_t [hipDeviceGetName](#) (char *name, int len, hipDevice_t device)
Returns an identifier string for the device.
- hipError_t [hipDeviceGetP2PAttribute](#) (int *value, hipDeviceP2PAttr attr, int srcDevice, int dstDevice)
Returns a value for attr of link between two devices.
- hipError_t [hipDeviceGetPCIBusId](#) (char *pciBusId, int len, int device)
Returns a PCI Bus Id string for the device, overloaded to take int device ID.
- hipError_t [hipDeviceGetByPCIBusId](#) (int *device, const char *pciBusId)
Returns a handle to a compute device.
- hipError_t [hipDeviceTotalMem](#) (size_t *bytes, hipDevice_t device)
Returns the total amount of memory on the device.

4.2.1 Detailed Description

This section describes the initialization and version functions of HIP runtime API.

4.2.2 Function Documentation

4.2.2.1 hipDeviceComputeCapability()

```
hipError_t hipDeviceComputeCapability (
    int * major,
    int * minor,
    hipDevice_t device )
```

Returns the compute capability of the device.

Parameters

out	<i>major</i>	
out	<i>minor</i>	
in	<i>device</i>	

Returns

#hipSuccess, #hipErrorInavlidDevice

4.2.2.2 hipDeviceGet()

```
hipError_t hipDeviceGet (
    hipDevice_t * device,
    int ordinal )
```

Returns a handle to a compute device.

Parameters

out	<i>device</i>	
in	<i>ordinal</i>	

Returns

#hipSuccess, #hipErrorInavlidDevice

4.2.2.3 hipDeviceGetByPCIBusId()

```
hipError_t hipDeviceGetByPCIBusId (
    int * device,
    const char * pciBusId )
```

Returns a handle to a compute device.

Parameters

out	<i>device</i>	handle
in	<i>PCI</i>	Bus ID

Returns

#hipSuccess, #hipErrorInavlidDevice, #hipErrorInvalidValue

4.2.2.4 hipDeviceGetName()

```
hipError_t hipDeviceGetName (
    char * name,
```

```
int len,  
hipDevice_t device )
```

Returns an identifier string for the device.

Parameters

out	<i>name</i>	
in	<i>len</i>	
in	<i>device</i>	

Returns

#hipSuccess, #hipErrorInavlidDevice

4.2.2.5 hipDeviceGetP2PAttribute()

```
hipError_t hipDeviceGetP2PAttribute (
    int * value,
    hipDeviceP2PAttr attr,
    int srcDevice,
    int dstDevice )
```

Returns a value for attr of link between two devices.

Parameters

out	<i>value</i>	
in	<i>attr</i>	
in	<i>srcDevice</i>	
in	<i>dstDevice</i>	

Returns

#hipSuccess, #hipErrorInavlidDevice

4.2.2.6 hipDeviceGetPCIBusId()

```
hipError_t hipDeviceGetPCIBusId (
    char * pciBusId,
    int len,
    int device )
```

Returns a PCI Bus Id string for the device, overloaded to take int device ID.

Parameters

out	<i>pci↔ BusId</i>	
in	<i>len</i>	
in	<i>device</i>	

Returns

#hipSuccess, #hipErrorInavlidDevice

4.2.2.7 hipDeviceTotalMem()

```
hipError_t hipDeviceTotalMem (
    size_t * bytes,
    hipDevice_t device )
```

Returns the total amount of memory on the device.

Parameters

out	<i>bytes</i>	
in	<i>device</i>	

Returns

#hipSuccess, #hipErrorInavlidDevice

4.2.2.8 hipDriverGetVersion()

```
hipError_t hipDriverGetVersion (
    int * driverVersion )
```

Returns the approximate HIP driver version.

Parameters

out	<i>driverVersion</i>	
-----	----------------------	--

Returns

#hipSuccess, #hipErrorInavlidValue

Warning

The HIP feature set does not correspond to an exact CUDA SDK driver revision. This function always set *driverVersion to 4 as an approximation though HIP supports some features which were introduced in later CUDA SDK revisions. HIP apps code should not rely on the driver revision number here and should use arch feature flags to test device capabilities or conditional compilation.

See also

[hipRuntimeGetVersion](#)

4.2.2.9 hipInit()

```
hipError_t hipInit (
    unsigned int flags )
```

Explicitly initializes the HIP runtime.

Most HIP APIs implicitly initialize the HIP runtime. This API provides control over the timing of the initialization.

4.2.2.10 hipRuntimeGetVersion()

```
hipError_t hipRuntimeGetVersion (
    int * runtimeVersion )
```

Returns the approximate HIP Runtime version.

Parameters

out	<i>runtimeVersion</i>	
-----	-----------------------	--

Returns

#hipSuccess, #hipErrorInvalidValue

Warning

On HIP/HCC path this function returns HIP runtime patch version however on HIP/NVCC path this function return CUDA runtime version.

See also

[hipDriverGetVersion](#)

4.3 Device Management

Functions

- `hipError_t hipDeviceSynchronize` (void)
Waits on all active streams on current device.
- `hipError_t hipDeviceReset` (void)
The state of current device is discarded and updated to a fresh state.
- `hipError_t hipSetDevice` (int deviceId)
Set default device to be used for subsequent hip API calls from this thread.
- `hipError_t hipGetDevice` (int *deviceId)
Return the default device id for the calling host thread.
- `hipError_t hipGetDeviceCount` (int *count)
Return number of compute-capable devices.
- `hipError_t hipDeviceGetAttribute` (int *pi, `hipDeviceAttribute_t` attr, int deviceId)
Query for a specific device attribute.
- `hipError_t hipGetDeviceProperties` (`hipDeviceProp_t` *prop, int deviceId)
Returns device properties.
- `hipError_t hipDeviceSetCacheConfig` (`hipFuncCache_t` cacheConfig)
Set L1/Shared cache partition.
- `hipError_t hipDeviceGetCacheConfig` (`hipFuncCache_t` *cacheConfig)
Set Cache configuration for a specific function.
- `hipError_t hipDeviceGetLimit` (size_t *pValue, enum `hipLimit_t` limit)
Get Resource limits of current device.
- `hipError_t hipDeviceGetSharedMemConfig` (`hipSharedMemConfig` *pConfig)
Returns bank width of shared memory for current device.
- `hipError_t hipGetDeviceFlags` (unsigned int *flags)
Gets the flags set for current device.
- `hipError_t hipDeviceSetSharedMemConfig` (`hipSharedMemConfig` config)
The bank width of shared memory on current device is set.
- `hipError_t hipSetDeviceFlags` (unsigned flags)
The current device behavior is changed according the flags passed.
- `hipError_t hipChooseDevice` (int *device, const `hipDeviceProp_t` *prop)
Device which matches `hipDeviceProp_t` is returned.
- `hipError_t hipExtGetLinkTypeAndHopCount` (int device1, int device2, uint32_t *linktype, uint32_t *hopcount)
Returns the link type and hop count between two devices.
- `hipError_t hipIpcGetMemHandle` (`hipIpcMemHandle_t` *handle, void *devPtr)
Gets an interprocess memory handle for an existing device memory allocation.
- `hipError_t hipIpcOpenMemHandle` (void **devPtr, `hipIpcMemHandle_t` handle, unsigned int flags)
Opens an interprocess memory handle exported from another process and returns a device pointer usable in the local process.
- `hipError_t hipIpcCloseMemHandle` (void *devPtr)
Close memory mapped with `hipIpcOpenMemHandle`.
- `hipError_t hipIpcGetEventHandle` (`hipIpcEventHandle_t` *handle, `hipEvent_t` event)
- `hipError_t hipIpcOpenEventHandle` (`hipEvent_t` *event, `hipIpcEventHandle_t` handle)

4.3.1 Detailed Description

This section describes the device management functions of HIP runtime API.

4.3.2 Function Documentation

4.3.2.1 hipChooseDevice()

```
hipError_t hipChooseDevice (
    int * device,
    const hipDeviceProp_t * prop )
```

Device which matches `hipDeviceProp_t` is returned.

Parameters

out	<i>device</i>	ID
in	<i>device</i>	properties pointer

Returns

#hipSuccess, #hipErrorInvalidValue

4.3.2.2 hipDeviceGetAttribute()

```
hipError_t hipDeviceGetAttribute (
    int * pi,
    hipDeviceAttribute_t attr,
    int deviceId )
```

Query for a specific device attribute.

Parameters

out	<i>pi</i>	pointer to value to return
in	<i>attr</i>	attribute to query
in	<i>deviceId</i>	which device to query for information

Returns

#hipSuccess, #hipErrorInvalidDevice, #hipErrorInvalidValue

4.3.2.3 hipDeviceGetCacheConfig()

```
hipError_t hipDeviceGetCacheConfig (
    hipFuncCache_t * cacheConfig )
```

Set Cache configuration for a specific function.

Parameters

in	<i>cacheConfig</i>	
----	--------------------	--

Returns

#hipSuccess, #hipErrorNotInitialized Note: AMD devices and some Nvidia GPUS do not support reconfigurable cache. This hint is ignored on those architectures.

4.3.2.4 hipDeviceGetLimit()

```
hipError_t hipDeviceGetLimit (
    size_t * pValue,
    enum hipLimit_t limit )
```

Get Resource limits of current device.

Parameters

out	<i>pValue</i>	
in	<i>limit</i>	

Returns

#hipSuccess, #hipErrorUnsupportedLimit, #hipErrorInvalidValue Note: Currently, only hipLimitMallocHeap↔ Size is available

4.3.2.5 hipDeviceGetSharedMemConfig()

```
hipError_t hipDeviceGetSharedMemConfig (
    hipSharedMemConfig * pConfig )
```

Returns bank width of shared memory for current device.

Parameters

out	<i>pConfig</i>	
-----	----------------	--

Returns

#hipSuccess, #hipErrorInvalidValue, #hipErrorNotInitialized

Note: AMD devices and some Nvidia GPUS do not support shared cache banking, and the hint is ignored on those architectures.

4.3.2.6 hipDeviceReset()

```
hipError_t hipDeviceReset (
    void )
```

The state of current device is discarded and updated to a fresh state.

Calling this function deletes all streams created, memory allocated, kernels running, events created. Make sure that no other thread is using the device or streams, memory, kernels, events associated with the current device.

Returns

#hipSuccess

See also

[hipDeviceSynchronize](#)

4.3.2.7 hipDeviceSetCacheConfig()

```
hipError_t hipDeviceSetCacheConfig (
    hipFuncCache_t cacheConfig )
```

Set L1/Shared cache partition.

Parameters

in	<i>cacheConfig</i>	
----	--------------------	--

Returns

#hipSuccess, #hipErrorNotInitialized Note: AMD devices and some Nvidia GPUS do not support reconfigurable cache. This hint is ignored on those architectures.

4.3.2.8 hipDeviceSetSharedMemConfig()

```
hipError_t hipDeviceSetSharedMemConfig (
    hipSharedMemConfig config )
```

The bank width of shared memory on current device is set.

Parameters

in	<i>config</i>	
----	---------------	--

Returns

#hipSuccess, #hipErrorInvalidValue, #hipErrorNotInitialized

Note: AMD devices and some Nvidia GPUS do not support shared cache banking, and the hint is ignored on those architectures.

4.3.2.9 hipDeviceSynchronize()

```
hipError_t hipDeviceSynchronize (
    void )
```

Waits on all active streams on current device.

When this command is invoked, the host thread gets blocked until all the commands associated with streams associated with the device. HIP does not support multiple blocking modes (yet!).

Returns

#hipSuccess

See also

[hipSetDevice](#), [hipDeviceReset](#)

4.3.2.10 hipExtGetLinkTypeAndHopCount()

```
hipError_t hipExtGetLinkTypeAndHopCount (
    int device1,
    int device2,
    uint32_t * linktype,
    uint32_t * hopcount )
```

Returns the link type and hop count between two devices.

Parameters

in	<i>device1</i>	Ordinal for device1
in	<i>device2</i>	Ordinal for device2
out	<i>linktype</i>	Returns the link type (See <code>hsa_amd_link_info_type_t</code>) between the two devices
out	<i>hopcount</i>	Returns the hop count between the two devices

Queries and returns the HSA link type and the hop count between the two specified devices.

Returns

#hipSuccess, #hipInvalidDevice, #hipErrorRuntimeOther

4.3.2.11 hipGetDevice()

```
hipError_t hipGetDevice (
    int * deviceId )
```

Return the default device id for the calling host thread.

Parameters

out	<i>device</i>	*device is written with the default device
-----	---------------	--

HIP maintains an default device for each thread using thread-local-storage. This device is used implicitly for HIP runtime APIs called by this thread. `hipGetDevice` returns in `*device` the default device for the calling host thread.

Returns

`#hipSuccess`, `#hipErrorInvalidDevice`, `#hipErrorInvalidValue`

See also

[hipSetDevice](#), `hipGetDevicesizeBytes`

4.3.2.12 hipGetDeviceCount()

```
hipError_t hipGetDeviceCount (
    int * count )
```

Return number of compute-capable devices.

Parameters

<i>[output]</i>	count Returns number of compute-capable devices.
-----------------	--

Returns

`#hipSuccess`, `#hipErrorNoDevice`

Returns in `*count` the number of devices that have ability to run compute commands. If there are no such devices, then [hipGetDeviceCount](#) will return `#hipErrorNoDevice`. If 1 or more devices can be found, then `hipGetDeviceCount` returns `#hipSuccess`.

4.3.2.13 hipGetDeviceFlags()

```
hipError_t hipGetDeviceFlags (
    unsigned int * flags )
```

Gets the flags set for current device.

Parameters

out	<i>flags</i>	
-----	--------------	--

Returns

#hipSuccess, #hipErrorInvalidDevice, #hipErrorInvalidValue

4.3.2.14 hipGetDeviceProperties()

```
hipError_t hipGetDeviceProperties (
    hipDeviceProp_t * prop,
    int deviceId )
```

Returns device properties.

Parameters

out	<i>prop</i>	written with device properties
in	<i>deviceId</i>	which device to query for information

Returns

#hipSuccess, #hipErrorInvalidDevice

Populates hipGetDeviceProperties with information for the specified device.

Parameters

out	<i>prop</i>	written with device properties
in	<i>deviceId</i>	which device to query for information

Returns

#hipSuccess, #hipErrorInvalidDevice

Populates hipGetDeviceProperties with information for the specified device.

4.3.2.15 hipIpcCloseMemHandle()

```
hipError_t hipIpcCloseMemHandle (
    void * devPtr )
```

Close memory mapped with hipIpcOpenMemHandle.

Unmaps memory returned by `hipIpcOpenMemHandle`. The original allocation in the exporting process as well as imported mappings in other processes will be unaffected.

Any resources used to enable peer access will be freed if this is the last mapping using them.

Parameters

<i>devPtr</i>	- Device pointer returned by <code>hipIpcOpenMemHandle</code>
---------------	---

Returns

`hipSuccess`, `hipErrorMapFailed`, `hipErrorInvalidHandle`,

4.3.2.16 hipIpcGetMemHandle()

```
hipError_t hipIpcGetMemHandle (
    hipIpcMemHandle_t * handle,
    void * devPtr )
```

Gets an interprocess memory handle for an existing device memory allocation.

Takes a pointer to the base of an existing device memory allocation created with `hipMalloc` and exports it for use in another process. This is a lightweight operation and may be called multiple times on an allocation without adverse effects.

If a region of memory is freed with `hipFree` and a subsequent call to `hipMalloc` returns memory with the same device address, `hipIpcGetMemHandle` will return a unique handle for the new memory.

Parameters

<i>handle</i>	- Pointer to user allocated <code>hipIpcMemHandle</code> to return the handle in.
<i>devPtr</i>	- Base pointer to previously allocated device memory

Returns

`hipSuccess`, `hipErrorInvalidHandle`, `hipErrorOutOfMemory`, `hipErrorMapFailed`,

4.3.2.17 hipIpcOpenMemHandle()

```
hipError_t hipIpcOpenMemHandle (
    void ** devPtr,
    hipIpcMemHandle_t handle,
    unsigned int flags )
```

Opens an interprocess memory handle exported from another process and returns a device pointer usable in the local process.

Maps memory exported from another process with `hipIpcGetMemHandle` into the current device address space. For contexts on different devices `hipIpcOpenMemHandle` can attempt to enable peer access between the devices as if the user called `hipDeviceEnablePeerAccess`. This behavior is controlled by the `hipIpcMemLazyEnablePeerAccess` flag. `hipDeviceCanAccessPeer` can determine if a mapping is possible.

Contexts that may open `hipIpcMemHandles` are restricted in the following way. `hipIpcMemHandles` from each device in a given process may only be opened by one context per device per other process.

Memory returned from `hipIpcOpenMemHandle` must be freed with `hipIpcCloseMemHandle`.

Calling `hipFree` on an exported memory region before calling `hipIpcCloseMemHandle` in the importing context will result in undefined behavior.

Parameters

<i>devPtr</i>	- Returned device pointer
<i>handle</i>	- <code>hipIpcMemHandle</code> to open
<i>flags</i>	- Flags for this operation. Must be specified as <code>hipIpcMemLazyEnablePeerAccess</code>

Returns

`hipSuccess`, `hipErrorMapFailed`, `hipErrorInvalidHandle`, `hipErrorTooManyPeers`

Note

No guarantees are made about the address returned in `*devPtr`. In particular, multiple processes may not receive the same address for the same `handle`.

4.3.2.18 hipSetDevice()

```
hipError_t hipSetDevice (
    int deviceId )
```

Set default device to be used for subsequent hip API calls from this thread.

Parameters

in	<i>deviceId</i>	Valid device in range 0... hipGetDeviceCount() .
----	-----------------	--

Sets `device` as the default device for the calling host thread. Valid device id's are 0... ([hipGetDeviceCount\(\)](#)-1).

Many HIP APIs implicitly use the "default device" :

- Any device memory subsequently allocated from this host thread (using `hipMalloc`) will be allocated on device.
- Any streams or events created from this host thread will be associated with device.
- Any kernels launched from this host thread (using `hipLaunchKernel`) will be executed on device (unless a specific stream is specified, in which case the device associated with that stream will be used).

This function may be called from any host thread. Multiple host threads may use the same device. This function does no synchronization with the previous or new device, and has very little runtime overhead. Applications can use `hipSetDevice` to quickly switch the default device before making a HIP runtime call which uses the default device.

The default device is stored in thread-local-storage for each thread. Thread-pool implementations may inherit the default device of the previous thread. A good practice is to always call `hipSetDevice` at the start of HIP coding sequency to establish a known standard device.

Returns

#hipSuccess, #hipErrorInvalidDevice, #hipErrorDeviceAlreadyInUse

See also

[hipGetDevice](#), [hipGetDeviceCount](#)

4.3.2.19 hipSetDeviceFlags()

```
hipError_t hipSetDeviceFlags (
    unsigned flags )
```

The current device behavior is changed according the flags passed.

Parameters

in	flags	
		The schedule flags impact how HIP waits for the completion of a command running on a device. hipDeviceScheduleSpin : HIP runtime will actively spin in the thread which submitted the work until the command completes. This offers the lowest latency, but will consume a CPU core and may increase power. hipDeviceScheduleYield : The HIP runtime will yield the CPU to system so that other tasks can use it. This may increase latency to detect the completion but will consume less power and is friendlier to other tasks in the system. hipDeviceScheduleBlockingSync : On ROCm platform, this is a synonym for hipDeviceScheduleYield. hipDeviceScheduleAuto : Use a heuristic to select between Spin and Yield modes. If the number of HIP contexts is greater than the number of logical processors in the system, use Spin scheduling. Else use Yield scheduling.

hipDeviceMapHost : Allow mapping host memory. On ROCM, this is always allowed and the flag is ignored. hip↔DeviceLmemResizeToMax :

Warning

ROCM silently ignores this flag.

Returns

#hipSuccess, #hipErrorInvalidDevice, #hipErrorSetActiveProcess

4.4 Execution Control

Functions

- `hipError_t hipFuncSetAttribute` (`const void *func`, `hipFuncAttribute attr`, `int value`)
Set attribute for a specific function.
- `hipError_t hipFuncSetCacheConfig` (`const void *func`, `hipFuncCache_t config`)
Set Cache configuration for a specific function.
- `hipError_t hipFuncSetSharedMemConfig` (`const void *func`, `hipSharedMemConfig config`)
Set shared memory configuration for a specific function.

4.4.1 Detailed Description

This section describes the execution control functions of HIP runtime API.

4.4.2 Function Documentation

4.4.2.1 hipFuncSetAttribute()

```
hipError_t hipFuncSetAttribute (
    const void * func,
    hipFuncAttribute attr,
    int value )
```

Set attribute for a specific function.

Parameters

in	<i>func</i> ;	
in	<i>attr</i> ;	
in	<i>value</i> ;	

Returns

`#hipSuccess`, `#hipErrorInvalidDeviceFunction`, `#hipErrorInvalidValue`

Note: AMD devices and some Nvidia GPUS do not support shared cache banking, and the hint is ignored on those architectures.

4.4.2.2 hipFuncSetCacheConfig()

```
hipError_t hipFuncSetCacheConfig (
    const void * func,
    hipFuncCache_t config )
```

Set Cache configuration for a specific function.

Parameters

in	<i>config</i> ;	
----	-----------------	--

Returns

#hipSuccess, #hipErrorNotInitialized Note: AMD devices and some Nvidia GPUS do not support reconfigurable cache. This hint is ignored on those architectures.

4.4.2.3 hipFuncSetSharedMemConfig()

```
hipError_t hipFuncSetSharedMemConfig (
    const void * func,
    hipSharedMemConfig config )
```

Set shared memory configuration for a specific function.

Parameters

in	<i>func</i>	
in	<i>config</i>	

Returns

#hipSuccess, #hipErrorInvalidDeviceFunction, #hipErrorInvalidValue

Note: AMD devices and some Nvidia GPUS do not support shared cache banking, and the hint is ignored on those architectures.

4.5 Error Handling

Functions

- `hipError_t hipGetLastError` (void)
Return last error returned by any HIP runtime API call and resets the stored error code to #hipSuccess.
- `hipError_t hipPeekAtLastError` (void)
Return last error returned by any HIP runtime API call.
- `const char * hipGetErrorName` (hipError_t hip_error)
Return name of the specified error code in text form.
- `const char * hipGetErrorString` (hipError_t hipError)
Return handy text string message to explain the error which occurred.

4.5.1 Detailed Description

This section describes the error handling functions of HIP runtime API.

4.5.2 Function Documentation

4.5.2.1 hipGetErrorName()

```
const char * hipGetErrorName (
    hipError_t hip_error )
```

Return name of the specified error code in text form.

Parameters

<i>hip_error</i>	Error code to convert to name.
------------------	--------------------------------

Returns

const char pointer to the NULL-terminated error name

See also

[hipGetErrorString](#), [hipGetLastError](#), [hipPeekAtLastError](#), [hipError_t](#)

4.5.2.2 hipGetErrorString()

```
const char * hipGetErrorString (
    hipError_t hipError )
```

Return handy text string message to explain the error which occurred.

Parameters

<i>hipError</i>	Error code to convert to string.
-----------------	----------------------------------

Returns

const char pointer to the NULL-terminated error string

Warning

: on HCC, this function returns the name of the error (same as `hipGetErrorName`)

See also

[hipGetErrorName](#), [hipGetLastError](#), [hipPeakAtLastError](#), [hipError_t](#)

4.5.2.3 hipGetLastError()

```
hipError_t hipGetLastError (
    void )
```

Return last error returned by any HIP runtime API call and resets the stored error code to `#hipSuccess`.

Returns

return code from last HIP called from the active host thread

Returns the last error that has been returned by any of the runtime calls in the same host thread, and then resets the saved error to `#hipSuccess`.

See also

[hipGetErrorString](#), [hipGetLastError](#), [hipPeakAtLastError](#), [hipError_t](#)

4.5.2.4 hipPeekAtLastError()

```
hipError_t hipPeekAtLastError (
    void )
```

Return last error returned by any HIP runtime API call.

Returns

`#hipSuccess`

Returns the last error that has been returned by any of the runtime calls in the same host thread. Unlike `hipGetLastError`, this function does not reset the saved error code.

See also

[hipGetErrorString](#), [hipGetLastError](#), [hipPeakAtLastError](#), [hipError_t](#)

4.6 Stream Management

Typedefs

- typedef void(* [hipStreamCallback_t](#)) (hipStream_t stream, hipError_t status, void *userData)
- typedef void(* [hipStreamCallback_t](#)) (hipStream_t stream, hipError_t status, void *userData)

Functions

- hipError_t [hipStreamCreate](#) (hipStream_t *stream)
Create an asynchronous stream.
- hipError_t [hipStreamCreateWithFlags](#) (hipStream_t *stream, unsigned int flags)
Create an asynchronous stream.
- hipError_t [hipStreamCreateWithPriority](#) (hipStream_t *stream, unsigned int flags, int priority)
Create an asynchronous stream with the specified priority.
- hipError_t [hipDeviceGetStreamPriorityRange](#) (int *leastPriority, int *greatestPriority)
Returns numerical values that correspond to the least and greatest stream priority.
- hipError_t [hipStreamDestroy](#) (hipStream_t stream)
Destroys the specified stream.
- hipError_t [hipStreamQuery](#) (hipStream_t stream)
Return #hipSuccess if all of the operations in the specified stream have completed, or #hipErrorNotReady if not.
- hipError_t [hipStreamSynchronize](#) (hipStream_t stream)
Wait for all commands in stream to complete.
- hipError_t [hipStreamWaitEvent](#) (hipStream_t stream, hipEvent_t event, unsigned int flags)
Make the specified compute stream wait for an event.
- hipError_t [hipStreamGetFlags](#) (hipStream_t stream, unsigned int *flags)
Return flags associated with this stream.
- hipError_t [hipStreamGetPriority](#) (hipStream_t stream, int *priority)
Query the priority of a stream.
- hipError_t [hipExtStreamCreateWithCUMask](#) (hipStream_t *stream, uint32_t cuMaskSize, const uint32_t *cuMask)
Create an asynchronous stream with the specified CU mask.
- hipError_t [hipExtStreamGetCUMask](#) (hipStream_t stream, uint32_t cuMaskSize, uint32_t *cuMask)
Get CU mask associated with an asynchronous stream.
- hipError_t [hipStreamAddCallback](#) (hipStream_t stream, [hipStreamCallback_t](#) callback, void *userData, unsigned int flags)
Adds a callback to be called on the host after all currently enqueued items in the stream have completed. For each cudaStreamAddCallback call, a callback will be executed exactly once. The callback will block later work in the stream until it is finished.
- hipError_t [hipStreamWaitValue32](#) (hipStream_t stream, void *ptr, int32_t value, unsigned int flags, uint32_t mask __dparm(0xFFFFFFFF))
Enqueues a wait command to the stream.
- hipError_t [hipStreamWaitValue64](#) (hipStream_t stream, void *ptr, int64_t value, unsigned int flags, uint64_t mask __dparm(0xFFFFFFFFFFFFFFFF))
Enqueues a wait command to the stream.
- hipError_t [hipStreamWriteValue32](#) (hipStream_t stream, void *ptr, int32_t value, unsigned int flags)
Enqueues a write command to the stream.
- hipError_t [hipStreamWriteValue64](#) (hipStream_t stream, void *ptr, int64_t value, unsigned int flags)
Enqueues a write command to the stream.

4.6.1 Detailed Description

This section describes the stream management functions of HIP runtime API. The following Stream APIs are not (yet) supported in HIP:

- `cudaStreamAttachMemAsync`

This section describes Stream Memory Wait and Write functions of HIP runtime API.

4.6.2 Typedef Documentation

```
typedef void(*hipStreamCallback_t) (hipStream_t stream, hipError_t status, void*userData)
```

Stream CallBack struct

4.6.3 Function Documentation

4.6.3.1 `hipDeviceGetStreamPriorityRange()`

```
hipError_t hipDeviceGetStreamPriorityRange (
```

```
    int *leastPriority,
    int *greatestPriority)
```

Returns numerical values that correspond to the least and greatest stream priority.

Parameters

in, out	<i>leastPriority</i>	pointer in which value corresponding to least priority is returned.
in, out	<i>greatestPriority</i>	pointer in which value corresponding to greatest priority is returned.

Returns in **leastPriority* and **greatestPriority* the numerical values that correspond to the least and greatest stream priority respectively. Stream priorities follow a convention where lower numbers imply greater priorities. The range of meaningful stream priorities is given by [**greatestPriority*, **leastPriority*]. If the user attempts to create a stream with a priority value that is outside the the meaningful range as specified by this API, the priority is automatically clamped to within the valid range.

4.6.3.2 `hipExtStreamCreateWithCUMask()`

```
hipError_t hipExtStreamCreateWithCUMask (
    hipStream_t * stream,
    uint32_t cuMaskSize,
    const uint32_t * cuMask )
```

Create an asynchronous stream with the specified CU mask.

Parameters

in, out	<i>stream</i>	Pointer to new stream
in	<i>cuMaskSize</i>	Size of CU mask bit array passed in.

Parameters

in	<i>cuMask</i>	Bit-vector representing the CU mask. Each active bit represents using one CU. The first 32 bits represent the first 32 CUs, and so on. If its size is greater than physical CU number (i.e., <code>multiProcessorCount</code> member of hipDeviceProp_t), the extra elements are ignored. It is user's responsibility to make sure the input is meaningful.
----	---------------	--

Returns

`#hipSuccess`, `#hipErrorInvalidHandle`, `#hipErrorInvalidValue`

Create a new asynchronous stream with the specified CU mask. `stream` returns an opaque handle that can be used to reference the newly created stream in subsequent `hipStream*` commands. The stream is allocated on the heap and will remain allocated even if the handle goes out-of-scope. To release the memory used by the stream, application must call `hipStreamDestroy`.

See also

[hipStreamCreate](#), [hipStreamSynchronize](#), [hipStreamWaitEvent](#), [hipStreamDestroy](#)

4.6.3.3 `hipExtStreamGetCUMask()`

```
hipError_t hipExtStreamGetCUMask (
    hipStream_t stream,
    uint32_t cuMaskSize,
    uint32_t * cuMask )
```

Get CU mask associated with an asynchronous stream.

Parameters

in	<i>stream</i>	stream to be queried
in	<i>cuMaskSize</i>	number of the block of memories (<code>uint32_t *</code>) allocated by user
out	<i>cuMask</i>	Pointer to a pre-allocated block of memories (<code>uint32_t *</code>) in which the stream's CU mask is returned. The CU mask is returned in a chunk of 32 bits where each active bit represents one active CU

Returns

`#hipSuccess`, `#hipErrorInvalidHandle`, `#hipErrorInvalidValue`

See also

[hipStreamCreate](#), [hipStreamSynchronize](#), [hipStreamWaitEvent](#), [hipStreamDestroy](#)

4.6.3.4 `hipStreamAddCallback()`

```
hipError_t hipStreamAddCallback (
    hipStream_t stream,
    hipStreamCallback_t callback,
    void * userData,
    unsigned int flags )
```

Adds a callback to be called on the host after all currently enqueued items in the stream have completed. For each `cudaStreamAddCallback` call, a callback will be executed exactly once. The callback will block later work in the stream until it is finished.

Parameters

in	<i>stream</i>	- Stream to add callback to
in	<i>callback</i>	- The function to call once preceding stream operations are complete
in	<i>userData</i>	- User specified data to be passed to the callback function
in	<i>flags</i>	- Reserved for future use, must be 0

Returns

#hipSuccess, #hipErrorInvalidHandle, #hipErrorNotSupported

See also

[hipStreamCreate](#), [hipStreamCreateWithFlags](#), [hipStreamQuery](#), [hipStreamSynchronize](#), [hipStreamWaitEvent](#), [hipStreamDestroy](#), [hipStreamCreateWithPriority](#)

4.6.3.5 hipStreamCreate()

```
hipError_t hipStreamCreate (
    hipStream_t * stream )
```

Create an asynchronous stream.

Parameters

in, out	<i>stream</i>	Valid pointer to hipStream_t. This function writes the memory with the newly created stream.
---------	---------------	--

Returns

#hipSuccess, #hipErrorInvalidValue

Create a new asynchronous stream. *stream* returns an opaque handle that can be used to reference the newly created stream in subsequent hipStream* commands. The stream is allocated on the heap and will remain allocated even if the handle goes out-of-scope. To release the memory used by the stream, applicaiton must call hipStream↵Destroy.

Returns

#hipSuccess, #hipErrorInvalidValue

See also

[hipStreamCreateWithFlags](#), [hipStreamCreateWithPriority](#), [hipStreamSynchronize](#), [hipStreamWaitEvent](#), [hipStreamDestroy](#)

4.6.3.6 hipStreamCreateWithFlags()

```
hipError_t hipStreamCreateWithFlags (
    hipStream_t * stream,
    unsigned int flags )
```

Create an asynchronous stream.

Parameters

in, out	<i>stream</i>	Pointer to new stream
in	<i>flags</i>	to control stream creation.

Returns

#hipSuccess, #hipErrorInvalidValue

Create a new asynchronous stream. `stream` returns an opaque handle that can be used to reference the newly created stream in subsequent `hipStream*` commands. The stream is allocated on the heap and will remain allocated even if the handle goes out-of-scope. To release the memory used by the stream, applicaiton must call `hipStream↔Destroy`. `Flags` controls behavior of the stream. See [hipStreamDefault](#), [hipStreamNonBlocking](#).

See also

[hipStreamCreate](#), [hipStreamCreateWithPriority](#), [hipStreamSynchronize](#), [hipStreamWaitEvent](#), [hipStreamDestroy](#)

4.6.3.7 hipStreamCreateWithPriority()

```
hipError_t hipStreamCreateWithPriority (
    hipStream_t * stream,
    unsigned int flags,
    int priority )
```

Create an asynchronous stream with the specified priority.

Parameters

in, out	<i>stream</i>	Pointer to new stream
in	<i>flags</i>	to control stream creation.
in	<i>priority</i>	of the stream. Lower numbers represent higher priorities.

Returns

#hipSuccess, #hipErrorInvalidValue

Create a new asynchronous stream with the specified priority. `stream` returns an opaque handle that can be used to reference the newly created stream in subsequent `hipStream*` commands. The stream is allocated on the heap and will remain allocated even if the handle goes out-of-scope. To release the memory used by the stream, applicaiton must call `hipStreamDestroy`. `Flags` controls behavior of the stream. See [hipStreamDefault](#), [hipStreamNonBlocking](#).

See also

[hipStreamCreate](#), [hipStreamSynchronize](#), [hipStreamWaitEvent](#), [hipStreamDestroy](#)

4.6.3.8 hipStreamDestroy()

```
hipError_t hipStreamDestroy (
    hipStream_t stream )
```

Destroys the specified stream.

Parameters

in, out	<i>stream</i>	Valid pointer to <code>hipStream_t</code> . This function writes the memory with the newly created stream.
---------	---------------	--

Returns

#hipSuccess #hipErrorInvalidHandle

Destroys the specified stream.

If commands are still executing on the specified stream, some may complete execution before the queue is deleted.

The queue may be destroyed while some commands are still inflight, or may wait for all commands queued to the stream before destroying it.

See also

[hipStreamCreate](#), [hipStreamCreateWithFlags](#), [hipStreamCreateWithPriority](#), [hipStreamQuery](#), [hipStreamWaitEvent](#), [hipStreamSynchronize](#)

4.6.3.9 hipStreamGetFlags()

```
hipError_t hipStreamGetFlags (
    hipStream_t stream,
    unsigned int * flags )
```

Return flags associated with this stream.

Parameters

in	<i>stream</i>	stream to be queried
in, out	<i>flags</i>	Pointer to an unsigned integer in which the stream's flags are returned

Returns

#hipSuccess, #hipErrorInvalidValue, #hipErrorInvalidHandle

#hipSuccess #hipErrorInvalidValue #hipErrorInvalidHandle

Return flags associated with this stream in *flags.

See also

[hipStreamCreateWithFlags](#)

4.6.3.10 hipStreamGetPriority()

```
hipError_t hipStreamGetPriority (
    hipStream_t stream,
    int * priority )
```

Query the priority of a stream.

Parameters

in	<i>stream</i>	stream to be queried
in, out	<i>priority</i>	Pointer to an unsigned integer in which the stream's priority is returned

Returns

#hipSuccess, #hipErrorInvalidValue, #hipErrorInvalidHandle

#hipSuccess #hipErrorInvalidValue #hipErrorInvalidHandle

Query the priority of a stream. The priority is returned in in priority.

See also

[hipStreamCreateWithFlags](#)

4.6.3.11 hipStreamQuery()

```
hipError_t hipStreamQuery (
    hipStream_t stream )
```

Return #hipSuccess if all of the operations in the specified `stream` have completed, or #hipErrorNotReady if not.

Parameters

in	<i>stream</i>	stream to query
----	---------------	-----------------

Returns

#hipSuccess, #hipErrorNotReady, #hipErrorInvalidHandle

This is thread-safe and returns a snapshot of the current state of the queue. However, if other host threads are sending work to the stream, the status may change immediately after the function is called. It is typically used for debug.

See also

[hipStreamCreate](#), [hipStreamCreateWithFlags](#), [hipStreamCreateWithPriority](#), [hipStreamWaitEvent](#), [hipStreamSynchronize](#), [hipStreamDestroy](#)

4.6.3.12 hipStreamSynchronize()

```
hipError_t hipStreamSynchronize (
    hipStream_t stream )
```

Wait for all commands in stream to complete.

Parameters

in	<i>stream</i>	stream identifier.
----	---------------	--------------------

Returns

#hipSuccess, #hipErrorInvalidHandle

This command is host-synchronous : the host will block until the specified stream is empty.

This command follows standard null-stream semantics. Specifically, specifying the null stream will cause the command to wait for other streams on the same device to complete all pending operations.

This command honors the `hipDeviceLaunchBlocking` flag, which controls whether the wait is active or blocking.

See also

[hipStreamCreate](#), [hipStreamCreateWithFlags](#), [hipStreamCreateWithPriority](#), [hipStreamWaitEvent](#), [hipStreamDestroy](#)

4.6.3.13 hipStreamWaitEvent()

```
hipError_t hipStreamWaitEvent (
    hipStream_t stream,
    hipEvent_t event,
    unsigned int flags )
```

Make the specified compute stream wait for an event.

Parameters

in	<i>stream</i>	stream to make wait.
in	<i>event</i>	event to wait on
in	<i>flags</i>	control operation [must be 0]

Returns

#hipSuccess, #hipErrorInvalidHandle

This function inserts a wait operation into the specified stream. All future work submitted to `stream` will wait until `event` reports completion before beginning execution.

This function only waits for commands in the current stream to complete. Notably, this function does not implicitly wait for commands in the default stream to complete, even if the specified stream is created with `hipStreamNonBlocking = 0`.

See also

[hipStreamCreate](#), [hipStreamCreateWithFlags](#), [hipStreamCreateWithPriority](#), [hipStreamSynchronize](#), [hipStreamDestroy](#)

4.6.3.14 hipStreamWaitValue32()

```
hipError_t hipStreamWaitValue32 (
    hipStream_t stream,
    void * ptr,
    int32_t value,
    unsigned int flags,
    uint32_t mask __dparm(0xFFFFFFFF) )
```

Enqueues a wait command to the stream.

Parameters

in	<i>stream</i>	- Stream identifier
in	<i>ptr</i>	- Pointer to memory object allocated using 'hipMallocSignalMemory' flag
in	<i>value</i>	- Value to be used in compare operation
in	<i>flags</i>	- Defines the compare operation, supported values are <code>hipStreamWaitValueGte</code> , <code>hipStreamWaitValueEq</code> , <code>hipStreamWaitValueAnd</code> and <code>hipStreamWaitValueNor</code>
in	<i>mask</i>	- Mask to be applied on value at memory before it is compared with value, default value is set to enable every bit

Returns

#hipSuccess, #hipErrorInvalidValue

Enqueues a wait command to the stream, all operations enqueued on this stream after this, will not execute until the defined wait condition is true.

`hipStreamWaitValueGte`: waits until `*ptr&mask >= value` `hipStreamWaitValueEq` : waits until `*ptr&mask == value`
`hipStreamWaitValueAnd`: waits until `((*ptr&mask) & value) != 0` `hipStreamWaitValueNor`: waits until `~((*ptr&mask) | (value&mask)) != 0`

Note

when using 'hipStreamWaitValueNor', mask is applied on both 'value' and '*ptr'.

Support for `hipStreamWaitValue32` can be queried using '[hipDeviceGetAttribute\(\)](#)' and '`hipDeviceAttributeCanUseStreamWaitValue`' flag.

See also

[hipExtMallocWithFlags](#), [hipFree](#), [hipStreamWaitValue64](#), [hipStreamWriteValue64](#), [hipStreamWriteValue32](#), [hipDeviceGetAttribute](#)

4.6.3.15 hipStreamWaitValue64()

```
hipError_t hipStreamWaitValue64 (
    hipStream_t stream,
    void * ptr,
    int64_t value,
    unsigned int flags,
    uint64_t mask __dparm(0xFFFFFFFFFFFFFFFF) )
```

Enqueues a wait command to the stream.

Parameters

in	<i>stream</i>	- Stream identifier
in	<i>ptr</i>	- Pointer to memory object allocated using 'hipMallocSignalMemory' flag
in	<i>value</i>	- Value to be used in compare operation
in	<i>flags</i>	- Defines the compare operation, supported values are hipStreamWaitValueGte hipStreamWaitValueEq, hipStreamWaitValueAnd and hipStreamWaitValueNor.
in	<i>mask</i>	- Mask to be applied on value at memory before it is compared with value default value is set to enable every bit

Returns

#hipSuccess, #hipErrorInvalidValue

Enqueues a wait command to the stream, all operations enqueued on this stream after this, will not execute until the defined wait condition is true.

hipStreamWaitValueGte: waits until *ptr&mask >= value hipStreamWaitValueEq : waits until *ptr&mask == value
 hipStreamWaitValueAnd: waits until ((*ptr&mask) & value) != 0 hipStreamWaitValueNor: waits until ~((*ptr&mask) | (value&mask)) != 0

Note

when using 'hipStreamWaitValueNor', mask is applied on both 'value' and '*ptr'.

Support for hipStreamWaitValue64 can be queried using '[hipDeviceGetAttribute\(\)](#)' and 'hipDeviceAttributeCanUseStreamWaitValue' flag.

See also

[hipExtMallocWithFlags](#), [hipFree](#), [hipStreamWaitValue32](#), [hipStreamWriteValue64](#), [hipStreamWriteValue32](#), [hipDeviceGetAttribute](#)

4.6.3.16 hipStreamWriteValue32()

```
hipError_t hipStreamWriteValue32 (
    hipStream_t stream,
    void * ptr,
    int32_t value,
    unsigned int flags )
```

Enqueues a write command to the stream.

Parameters

in	<i>stream</i>	- Stream identifier
in	<i>ptr</i>	- Pointer to a GPU accessible memory object
in	<i>value</i>	- Value to be written
in	<i>flags</i>	- reserved, ignored for now, will be used in future releases

Returns

#hipSuccess, #hipErrorInvalidValue

Enqueues a write command to the stream, write operation is performed after all earlier commands on this stream have completed the execution.

See also

[hipExtMallocWithFlags](#), [hipFree](#), [hipStreamWriteValue32](#), [hipStreamWaitValue32](#), [hipStreamWaitValue64](#)

4.6.3.17 hipStreamWriteValue64()

```
hipError_t hipStreamWriteValue64 (
    hipStream_t stream,
    void * ptr,
    int64_t value,
    unsigned int flags )
```

Enqueues a write command to the stream.

Parameters

in	<i>stream</i>	- Stream identifier
in	<i>ptr</i>	- Pointer to a GPU accessible memory object
in	<i>value</i>	- Value to be written
in	<i>flags</i>	- reserved, ignored for now, will be used in future releases

Returns

#hipSuccess, #hipErrorInvalidValue

Enqueues a write command to the stream, write operation is performed after all earlier commands on this stream have completed the execution.

See also

[hipExtMallocWithFlags](#), [hipFree](#), [hipStreamWriteValue32](#), [hipStreamWaitValue32](#), [hipStreamWaitValue64](#)

4.7 Event Management

Functions

- `hipError_t hipEventCreateWithFlags` (`hipEvent_t *event`, unsigned flags)
Create an event with the specified flags.
- `hipError_t hipEventCreate` (`hipEvent_t *event`)
- `hipError_t hipEventRecord` (`hipEvent_t event`, `hipStream_t stream`)
Record an event in the specified stream.
- `hipError_t hipEventDestroy` (`hipEvent_t event`)
Destroy the specified event.
- `hipError_t hipEventSynchronize` (`hipEvent_t event`)
Wait for an event to complete.
- `hipError_t hipEventElapsedTime` (`float *ms`, `hipEvent_t start`, `hipEvent_t stop`)
Return the elapsed time between two events.
- `hipError_t hipEventQuery` (`hipEvent_t event`)
Query event status.

4.7.1 Detailed Description

This section describes the event management functions of HIP runtime API.

4.7.2 Function Documentation

4.7.2.1 hipEventCreate()

```
hipError_t hipEventCreate (
    hipEvent_t * event )
```

Create an event

Parameters

<code>in, out</code>	<code>event</code>	Returns the newly created event.
----------------------	--------------------	----------------------------------

Returns

`#hipSuccess`, `#hipErrorNotInitialized`, `#hipErrorInvalidValue`, `#hipErrorLaunchFailure`, `#hipErrorOutOfMemory`

See also

[hipEventCreateWithFlags](#), [hipEventRecord](#), [hipEventQuery](#), [hipEventSynchronize](#), [hipEventDestroy](#), [hipEventElapsedTime](#)

4.7.2.2 hipEventCreateWithFlags()

```
hipError_t hipEventCreateWithFlags (
    hipEvent_t * event,
    unsigned flags )
```

Create an event with the specified flags.

Parameters

<code>in, out</code>	<code>event</code>	Returns the newly created event.
<code>in</code>	<code>flags</code>	Flags to control event behavior. Valid values are hipEventDefault , hipEventBlockingSync , hipEventDisableTiming , hipEventInterprocess

[hipEventDefault](#) : Default flag. The event will use active synchronization and will support timing. Blocking synchronization provides lowest possible latency at the expense of dedicating a CPU to poll on the event. [hipEventBlockingSync](#) : The event will use blocking synchronization : if `hipEventSynchronize` is called on this event, the thread will block until the event completes. This can increase latency for the synchroniation but can result in lower power and more resources for other CPU threads. [hipEventDisableTiming](#) : Disable recording of timing information. Events created with this flag would not record profiling data and provide best performance if used for synchronization.

Warning

On AMD platform, `hipEventInterprocess` support is under development. Use of this flag will return an error.

Returns

`#hipSuccess`, `#hipErrorNotInitialized`, `#hipErrorInvalidValue`, `#hipErrorLaunchFailure`, `#hipErrorOutOfMemory`

See also

[hipEventCreate](#), [hipEventSynchronize](#), [hipEventDestroy](#), [hipEventElapsedTime](#)

4.7.2.3 hipEventDestroy()

```
hipError_t hipEventDestroy (
    hipEvent_t event )
```

Destroy the specified event.

Parameters

in	<i>event</i>	Event to destroy.
----	--------------	-------------------

Returns

`#hipSuccess`, `#hipErrorNotInitialized`, `#hipErrorInvalidValue`, `#hipErrorLaunchFailure`

Releases memory associated with the event. If the event is recording but has not completed recording when [hipEventDestroy\(\)](#) is called, the function will return immediately and the `completion_future` resources will be released later, when the `hipDevice` is synchronized.

See also

[hipEventCreate](#), [hipEventCreateWithFlags](#), [hipEventQuery](#), [hipEventSynchronize](#), [hipEventRecord](#), [hipEventElapsedTime](#)

Returns

`#hipSuccess`

4.7.2.4 hipEventElapsedTime()

```
hipError_t hipEventElapsedTime (
    float * ms,
    hipEvent_t start,
    hipEvent_t stop )
```

Return the elapsed time between two events.

Parameters

out	ms	: Return time between start and stop in ms.
in	start	: Start event.
in	stop	: Stop event.

Returns

#hipSuccess, #hipErrorInvalidValue, #hipErrorNotReady, #hipErrorInvalidHandle, #hipErrorNotInitialized, #hipErrorLaunchFailure

Computes the elapsed time between two events. Time is computed in ms, with a resolution of approximately 1 us. Events which are recorded in a NULL stream will block until all commands on all other streams complete execution, and then record the timestamp.

Events which are recorded in a non-NULL stream will record their timestamp when they reach the head of the specified stream, after all previous commands in that stream have completed executing. Thus the time that the event recorded may be significantly after the host calls [hipEventRecord\(\)](#).

If [hipEventRecord\(\)](#) has not been called on either event, then #hipErrorInvalidHandle is returned. If [hipEventRecord\(\)](#) has been called on both events, but the timestamp has not yet been recorded on one or both events (that is, [hipEventQuery\(\)](#) would return #hipErrorNotReady on at least one of the events), then #hipErrorNotReady is returned.

Note, for HIP Events used in kernel dispatch using [hipExtLaunchKernelGGL](#)/[hipExtLaunchKernel](#), events passed in [hipExtLaunchKernelGGL](#)/[hipExtLaunchKernel](#) are not explicitly recorded and should only be used to get elapsed time for that specific launch. In case events are used across multiple dispatches, for example, start and stop events from different [hipExtLaunchKernelGGL](#)/[hipExtLaunchKernel](#) calls, they will be treated as invalid unrecorded events, HIP will throw error "hipErrorInvalidHandle" from [hipEventElapsedTime](#).

See also

[hipEventCreate](#), [hipEventCreateWithFlags](#), [hipEventQuery](#), [hipEventDestroy](#), [hipEventRecord](#), [hipEventSynchronize](#)

4.7.2.5 [hipEventQuery\(\)](#)

```
hipError_t hipEventQuery (
    hipEvent_t event )
```

Query event status.

Parameters

in	event	Event to query.
----	-------	-----------------

Returns

#hipSuccess, #hipErrorNotReady, #hipErrorInvalidHandle, #hipErrorInvalidValue, #hipErrorNotInitialized, #hipErrorLaunchFailure

Query the status of the specified event. This function will return #hipErrorNotReady if all commands in the appropriate stream (specified to [hipEventRecord\(\)](#)) have completed. If that work has not completed, or if [hipEventRecord\(\)](#) was not called on the event, then #hipSuccess is returned.

See also

[hipEventCreate](#), [hipEventCreateWithFlags](#), [hipEventRecord](#), [hipEventDestroy](#), [hipEventSynchronize](#), [hipEventElapsedTime](#)

4.7.2.6 [hipEventRecord\(\)](#)

```
hipError_t hipEventRecord (
```

```
hipEvent_t event,
hipStream_t stream )
```

Record an event in the specified stream.

Parameters

in	<i>event</i>	event to record.
in	<i>stream</i>	stream in which to record event.

Returns

#hipSuccess, #hipErrorInvalidValue, #hipErrorNotInitialized, #hipErrorInvalidHandle, #hipErrorLaunchFailure

[hipEventQuery\(\)](#) or [hipEventSynchronize\(\)](#) must be used to determine when the event transitions from "recording" (after [hipEventRecord\(\)](#) is called) to "recorded" (when timestamps are set, if requested).

Events which are recorded in a non-NULL stream will transition to from recording to "recorded" state when they reach the head of the specified stream, after all previous commands in that stream have completed executing.

If [hipEventRecord\(\)](#) has been previously called on this event, then this call will overwrite any existing state in event.

If this function is called on an event that is currently being recorded, results are undefined

- either outstanding recording may save state into the event, and the order is not guaranteed.

See also

[hipEventCreate](#), [hipEventCreateWithFlags](#), [hipEventQuery](#), [hipEventSynchronize](#), [hipEventDestroy](#), [hipEventElapsedTime](#)

4.7.2.7 hipEventSynchronize()

```
hipError_t hipEventSynchronize (
    hipEvent_t event )
```

Wait for an event to complete.

This function will block until the event is ready, waiting for all previous work in the stream specified when event was recorded with [hipEventRecord\(\)](#).

If [hipEventRecord\(\)](#) has not been called on *event*, this function returns immediately.

TODO-hip- This function needs to support `hipEventBlockingSync` parameter.

Parameters

in	<i>event</i>	Event on which to wait.
----	--------------	-------------------------

Returns

#hipSuccess, #hipErrorInvalidValue, #hipErrorNotInitialized, #hipErrorInvalidHandle, #hipErrorLaunchFailure

See also

[hipEventCreate](#), [hipEventCreateWithFlags](#), [hipEventQuery](#), [hipEventDestroy](#), [hipEventRecord](#), [hipEventElapsedTime](#)

4.8 Memory Management

Modules

- [Managed Memory \(ROCm HMM\)](#)

Functions

- `hipError_t hipPointerGetAttributes (hipPointerAttribute_t *attributes, const void *ptr)`
Return attributes for the specified pointer.
- `hipError_t hipImportExternalMemory (hipExternalMemory_t *extMem_out, const hipExternalMemoryHandleDesc *memHandleDesc)`
Imports an external memory object.
- `hipError_t hipExternalMemoryGetMappedBuffer (void **devPtr, hipExternalMemory_t extMem, const hipExternalMemoryBufferDesc *bufferDesc)`
Maps a buffer onto an imported memory object.
- `hipError_t hipDestroyExternalMemory (hipExternalMemory_t extMem)`
Destroys an external memory object.
- `hipError_t hipMalloc (void **ptr, size_t size)`
Allocate memory on the default accelerator.
- `hipError_t hipExtMallocWithFlags (void **ptr, size_t sizeBytes, unsigned int flags)`
Allocate memory on the default accelerator.
- `hipError_t hipMallocHost (void **ptr, size_t size)`
Allocate pinned host memory [Deprecated].
- `hipError_t hipMemAllocHost (void **ptr, size_t size)`
Allocate pinned host memory [Deprecated].
- `hipError_t hipHostMalloc (void **ptr, size_t size, unsigned int flags)`
Allocate device accessible page locked host memory.
- `hipError_t hipHostAlloc (void **ptr, size_t size, unsigned int flags)`
Allocate device accessible page locked host memory [Deprecated].
- `hipError_t hipHostGetDevicePointer (void **devPtr, void *hstPtr, unsigned int flags)`
Get Device pointer from Host Pointer allocated through hipHostMalloc.
- `hipError_t hipHostGetFlags (unsigned int *flagsPtr, void *hostPtr)`
Return flags associated with host pointer.
- `hipError_t hipHostRegister (void *hostPtr, size_t sizeBytes, unsigned int flags)`
Register host memory so it can be accessed from the current device.
- `hipError_t hipHostUnregister (void *hostPtr)`
Un-register host pointer.
- `hipError_t hipMallocPitch (void **ptr, size_t *pitch, size_t width, size_t height)`
- `hipError_t hipMemAllocPitch (hipDeviceptr_t *dptr, size_t *pitch, size_t widthInBytes, size_t height, unsigned int elementSizeBytes)`
- `hipError_t hipFree (void *ptr)`
Free memory allocated by the hcc hip memory allocation API. This API performs an implicit [hipDeviceSynchronize\(\)](#) call. If pointer is NULL, the hip runtime is initialized and hipSuccess is returned.
- `hipError_t hipFreeHost (void *ptr)`
Free memory allocated by the hcc hip host memory allocation API. [Deprecated].
- `hipError_t hipHostFree (void *ptr)`
Free memory allocated by the hcc hip host memory allocation API This API performs an implicit [hipDeviceSynchronize\(\)](#) call. If pointer is NULL, the hip runtime is initialized and hipSuccess is returned.
- `hipError_t hipMemcpy (void *dst, const void *src, size_t sizeBytes, hipMemcpyKind kind)`
Copy data from src to dst.
- `hipError_t hipMemcpyWithStream (void *dst, const void *src, size_t sizeBytes, hipMemcpyKind kind, hipStream_t stream)`

- hipError_t [hipMemcpyHtoD](#) (hipDeviceptr_t dst, void *src, size_t sizeBytes)
Copy data from Host to Device.
- hipError_t [hipMemcpyDtoH](#) (void *dst, hipDeviceptr_t src, size_t sizeBytes)
Copy data from Device to Host.
- hipError_t [hipMemcpyDtoD](#) (hipDeviceptr_t dst, hipDeviceptr_t src, size_t sizeBytes)
Copy data from Device to Device.
- hipError_t [hipMemcpyHtoDAsync](#) (hipDeviceptr_t dst, void *src, size_t sizeBytes, hipStream_t stream)
Copy data from Host to Device asynchronously.
- hipError_t [hipMemcpyDtoHAsync](#) (void *dst, hipDeviceptr_t src, size_t sizeBytes, hipStream_t stream)
Copy data from Device to Host asynchronously.
- hipError_t [hipMemcpyDtoDAsync](#) (hipDeviceptr_t dst, hipDeviceptr_t src, size_t sizeBytes, hipStream_t stream)
Copy data from Device to Device asynchronously.
- hipError_t [hipModuleGetGlobal](#) (hipDeviceptr_t *dptr, size_t *bytes, hipModule_t hmod, const char *name)
- hipError_t [hipGetSymbolAddress](#) (void **devPtr, const void *symbol)
- hipError_t [hipGetSymbolSize](#) (size_t *size, const void *symbol)
- hipError_t [hipMemcpyToSymbol](#) (const void *symbol, const void *src, size_t sizeBytes, size_t offset, __dparm(0), hipMemcpyKind kind __dparm(hipMemcpyHostToDevice))
- hipError_t [hipMemcpyToSymbolAsync](#) (const void *symbol, const void *src, size_t sizeBytes, size_t offset, hipMemcpyKind kind, hipStream_t stream __dparm(0))
- hipError_t [hipMemcpyFromSymbol](#) (void *dst, const void *symbol, size_t sizeBytes, size_t offset, __dparm(0), hipMemcpyKind kind __dparm(hipMemcpyDeviceToHost))
- hipError_t [hipMemcpyFromSymbolAsync](#) (void *dst, const void *symbol, size_t sizeBytes, size_t offset, hipMemcpyKind kind, hipStream_t stream __dparm(0))
- hipError_t [hipMemcpyAsync](#) (void *dst, const void *src, size_t sizeBytes, hipMemcpyKind kind, hipStream_t stream __dparm(0))
Copy data from src to dst asynchronously.
- hipError_t [hipMemset](#) (void *dst, int value, size_t sizeBytes)
Fills the first sizeBytes bytes of the memory area pointed to by dest with the constant byte value value.
- hipError_t [hipMemsetD8](#) (hipDeviceptr_t dest, unsigned char value, size_t count)
Fills the first sizeBytes bytes of the memory area pointed to by dest with the constant byte value value.
- hipError_t [hipMemsetD8Async](#) (hipDeviceptr_t dest, unsigned char value, size_t count, hipStream_t stream __dparm(0))
Fills the first sizeBytes bytes of the memory area pointed to by dest with the constant byte value value.
- hipError_t [hipMemsetD16](#) (hipDeviceptr_t dest, unsigned short value, size_t count)
Fills the first sizeBytes bytes of the memory area pointed to by dest with the constant short value value.
- hipError_t [hipMemsetD16Async](#) (hipDeviceptr_t dest, unsigned short value, size_t count, hipStream_t stream __dparm(0))
Fills the first sizeBytes bytes of the memory area pointed to by dest with the constant short value value.
- hipError_t [hipMemsetD32](#) (hipDeviceptr_t dest, int value, size_t count)
Fills the memory area pointed to by dest with the constant integer value for specified number of times.
- hipError_t [hipMemsetAsync](#) (void *dst, int value, size_t sizeBytes, hipStream_t stream __dparm(0))
Fills the first sizeBytes bytes of the memory area pointed to by dev with the constant byte value value.
- hipError_t [hipMemsetD32Async](#) (hipDeviceptr_t dst, int value, size_t count, hipStream_t stream __dparm(0))
Fills the memory area pointed to by dev with the constant integer value for specified number of times.
- hipError_t [hipMemset2D](#) (void *dst, size_t pitch, int value, size_t width, size_t height)
Fills the memory area pointed to by dst with the constant value.
- hipError_t [hipMemset2DAsync](#) (void *dst, size_t pitch, int value, size_t width, size_t height, hipStream_t stream __dparm(0))
Fills asynchronously the memory area pointed to by dst with the constant value.
- hipError_t [hipMemset3D](#) ([hipPitchedPtr](#) pitchedDevPtr, int value, [hipExtent](#) extent)
Fills synchronously the memory area pointed to by pitchedDevPtr with the constant value.

- `hipError_t hipMemset3DAsync (hipPitchedPtr pitchedDevPtr, int value, hipExtent extent, hipStream_t stream __dparm(0))`
Fills asynchronously the memory area pointed to by pitchedDevPtr with the constant value.
- `hipError_t hipMemGetInfo (size_t *free, size_t *total)`
Query memory info. Return snapshot of free memory, and total allocatable memory on the device.
- `hipError_t hipMemPtrGetInfo (void *ptr, size_t *size)`
- `hipError_t hipMallocArray (hipArray **array, const hipChannelFormatDesc *desc, size_t width, size_t height __dparm(0), unsigned int flags __dparm(hipArrayDefault))`
Allocate an array on the device.
- `hipError_t hipArrayCreate (hipArray **pHandle, const HIP_ARRAY_DESCRIPTOR *pAllocateArray)`
- `hipError_t hipArrayDestroy (hipArray *array)`
- `hipError_t hipArray3DCreate (hipArray **array, const HIP_ARRAY3D_DESCRIPTOR *pAllocateArray)`
- `hipError_t hipMalloc3D (hipPitchedPtr *pitchedDevPtr, hipExtent extent)`
- `hipError_t hipFreeArray (hipArray *array)`
Frees an array on the device.
- `hipError_t hipFreeMipmappedArray (hipMipmappedArray_t mipmappedArray)`
Frees a mipmapped array on the device.
- `hipError_t hipMalloc3DArray (hipArray **array, const struct hipChannelFormatDesc *desc, struct hipExtent extent, unsigned int flags)`
Allocate an array on the device.
- `hipError_t hipMallocMipmappedArray (hipMipmappedArray_t *mipmappedArray, const struct hipChannelFormatDesc *desc, struct hipExtent extent, unsigned int numLevels, unsigned int flags __dparm(0))`
Allocate a mipmapped array on the device.
- `hipError_t hipGetMipmappedArrayLevel (hipArray_t *levelArray, hipMipmappedArray_const_t mipmappedArray, unsigned int level)`
Gets a mipmap level of a HIP mipmapped array.
- `hipError_t hipMemcpy2D (void *dst, size_t dpitch, const void *src, size_t spitch, size_t width, size_t height, hipMemcpyKind kind)`
Copies data between host and device.
- `hipError_t hipMemcpyParam2D (const hip_Memcpy2D *pCopy)`
Copies memory for 2D arrays.
- `hipError_t hipMemcpyParam2DAsync (const hip_Memcpy2D *pCopy, hipStream_t stream __dparm(0))`
Copies memory for 2D arrays.
- `hipError_t hipMemcpy2DAsync (void *dst, size_t dpitch, const void *src, size_t spitch, size_t width, size_t height, hipMemcpyKind kind, hipStream_t stream __dparm(0))`
Copies data between host and device.
- `hipError_t hipMemcpy2DToArray (hipArray *dst, size_t wOffset, size_t hOffset, const void *src, size_t spitch, size_t width, size_t height, hipMemcpyKind kind)`
Copies data between host and device.
- `hipError_t hipMemcpy2DToArrayAsync (hipArray *dst, size_t wOffset, size_t hOffset, const void *src, size_t spitch, size_t width, size_t height, hipMemcpyKind kind, hipStream_t stream __dparm(0))`
Copies data between host and device.
- `hipError_t hipMemcpyToArray (hipArray *dst, size_t wOffset, size_t hOffset, const void *src, size_t count, hipMemcpyKind kind)`
Copies data between host and device.
- `hipError_t hipMemcpyFromArray (void *dst, hipArray_const_t srcArray, size_t wOffset, size_t hOffset, size_t count, hipMemcpyKind kind)`
Copies data between host and device.
- `hipError_t hipMemcpy2DFromArray (void *dst, size_t dpitch, hipArray_const_t src, size_t wOffset, size_t hOffset, size_t width, size_t height, hipMemcpyKind kind)`
Copies data between host and device.
- `hipError_t hipMemcpy2DFromArrayAsync (void *dst, size_t dpitch, hipArray_const_t src, size_t wOffset, size_t hOffset, size_t width, size_t height, hipMemcpyKind kind, hipStream_t stream __dparm(0))`

Copies data between host and device asynchronously.

- hipError_t [hipMemcpyAtoH](#) (void *dst, [hipArray](#) *srcArray, size_t srcOffset, size_t count)

Copies data between host and device.

- hipError_t [hipMemcpyHtoA](#) ([hipArray](#) *dstArray, size_t dstOffset, const void *srcHost, size_t count)

Copies data between host and device.

- hipError_t [hipMemcpy3D](#) (const struct [hipMemcpy3DParms](#) *p)

Copies data between host and device.

- hipError_t [hipMemcpy3DAsync](#) (const struct [hipMemcpy3DParms](#) *p, hipStream_t stream __dparm(0))

Copies data between host and device asynchronously.

- hipError_t [hipDrvMemcpy3D](#) (const [HIP_MEMCPY3D](#) *pCopy)

Copies data between host and device.

- hipError_t [hipDrvMemcpy3DAsync](#) (const [HIP_MEMCPY3D](#) *pCopy, hipStream_t stream)

Copies data between host and device asynchronously.

4.8.1 Detailed Description

This section describes the memory management functions of HIP runtime API. The following CUDA APIs are not currently supported:

- cudaMalloc3D
- cudaMalloc3DArray
- TODO - more 2D, 3D, array APIs here.

4.8.2 Function Documentation

4.8.2.1 hipDestroyExternalMemory()

```
hipError_t hipDestroyExternalMemory (
    hipExternalMemory_t extMem )
```

Destroys an external memory object.

Parameters

in	<i>extMem</i>	External memory object to be destroyed
----	---------------	--

Returns

#hipSuccess, #hipErrorInvalidDevice, #hipErrorInvalidValue

See also

4.8.2.2 hipDrvMemcpy3D()

```
hipError_t hipDrvMemcpy3D (
    const HIP\_MEMCPY3D * pCopy )
```

Copies data between host and device.

Parameters

in	<i>pCopy</i>	3D memory copy parameters
----	--------------	---------------------------

Returns

#hipSuccess, #hipErrorInvalidValue, #hipErrorInvalidPitchValue, #hipErrorInvalidDevicePointer, #hipErrorInvalidMemcpyDirection

See also

[hipMemcpy](#), [hipMemcpy2DToArray](#), [hipMemcpy2D](#), [hipMemcpyFromArray](#), [hipMemcpyToSymbol](#), [hipMemcpyAsync](#)

4.8.2.3 hipDrvMemcpy3DAsync()

```
hipError_t hipDrvMemcpy3DAsync (
    const HIP_MEMCPY3D * pCopy,
    hipStream_t stream )
```

Copies data between host and device asynchronously.

Parameters

in	<i>pCopy</i>	3D memory copy parameters
in	<i>stream</i>	Stream to use

Returns

#hipSuccess, #hipErrorInvalidValue, #hipErrorInvalidPitchValue, #hipErrorInvalidDevicePointer, #hipErrorInvalidMemcpyDirection

See also

[hipMemcpy](#), [hipMemcpy2DToArray](#), [hipMemcpy2D](#), [hipMemcpyFromArray](#), [hipMemcpyToSymbol](#), [hipMemcpyAsync](#)

4.8.2.4 hipExternalMemoryGetMappedBuffer()

```
hipError_t hipExternalMemoryGetMappedBuffer (
    void ** devPtr,
    hipExternalMemory_t extMem,
    const hipExternalMemoryBufferDesc * bufferDesc )
```

Maps a buffer onto an imported memory object.

Parameters

out	<i>devPtr</i>	Returned device pointer to buffer
in	<i>extMem</i>	Handle to external memory object
in	<i>bufferDesc</i>	Buffer descriptor

Returns

#hipSuccess, #hipErrorInvalidDevice, #hipErrorInvalidValue

See also

4.8.2.5 hipExtMallocWithFlags()

```
hipError_t hipExtMallocWithFlags (
    void ** ptr,
    size_t sizeBytes,
    unsigned int flags )
```

Allocate memory on the default accelerator.

Parameters

out	<i>ptr</i>	Pointer to the allocated memory
in	<i>size</i>	Requested memory size
in	<i>flags</i>	Type of memory allocation

If size is 0, no memory is allocated, *ptr returns nullptr, and hipSuccess is returned.

Returns

#hipSuccess, #hipErrorOutOfMemory, #hipErrorInvalidValue (bad context, null *ptr)

See also

[hipMallocPitch](#), [hipFree](#), [hipMallocArray](#), [hipFreeArray](#), [hipMalloc3D](#), [hipMalloc3DArray](#), [hipHostFree](#), [hipHostMalloc](#)

4.8.2.6 hipFree()

```
hipError_t hipFree (
    void * ptr )
```

Free memory allocated by the hcc hip memory allocation API. This API performs an implicit [hipDeviceSynchronize\(\)](#) call. If pointer is NULL, the hip runtime is initialized and hipSuccess is returned.

Parameters

in	<i>ptr</i>	Pointer to memory to be freed
----	------------	-------------------------------

Returns

#hipSuccess

#hipErrorInvalidDevicePointer (if pointer is invalid, including host pointers allocated with hipHostMalloc)

See also

[hipMalloc](#), [hipMallocPitch](#), [hipMallocArray](#), [hipFreeArray](#), [hipHostFree](#), [hipMalloc3D](#), [hipMalloc3DArray](#), [hipHostMalloc](#)

4.8.2.7 hipFreeArray()

```
hipError_t hipFreeArray (
    hipArray * array )
```

Frees an array on the device.

Parameters

in	<i>array</i>	Pointer to array to free
----	--------------	--------------------------

Returns

#hipSuccess, #hipErrorInvalidValue, #hipErrorNotInitialized

See also

[hipMalloc](#), [hipMallocPitch](#), [hipFree](#), [hipMallocArray](#), [hipHostMalloc](#), [hipHostFree](#)

4.8.2.8 hipFreeHost()

```
static hipError_t hipFreeHost (
    void * ptr ) [inline]
```

Free memory allocated by the hcc hip host memory allocation API. [Deprecated].

Parameters

in	<i>ptr</i>	Pointer to memory to be freed
----	------------	-------------------------------

Returns

#hipSuccess, #hipErrorInvalidValue (if pointer is invalid, including device pointers allocated with hipMalloc)

Parameters

in	<i>ptr</i>	Pointer to memory to be freed
----	------------	-------------------------------

Returns

#hipSuccess, #hipErrorInvalidValue (if pointer is invalid, including device pointers allocated with hipMalloc)

4.8.2.9 hipFreeMipmappedArray()

```
hipError_t hipFreeMipmappedArray (
    hipMipmappedArray_t mipmappedArray )
```

Frees a mipmapped array on the device.

Parameters

in	<i>mipmappedArray</i>	- Pointer to mipmapped array to free
----	-----------------------	--------------------------------------

Returns

#hipSuccess, #hipErrorInvalidValue

4.8.2.10 hipGetMipmappedArrayLevel()

```
hipError_t hipGetMipmappedArrayLevel (
    hipArray_t * levelArray,
    hipMipmappedArray_const_t mipmappedArray,
    unsigned int level )
```

Gets a mipmap level of a HIP mipmapped array.

Parameters

out	<i>levelArray</i>	- Returned mipmap level HIP array
in	<i>mipmappedArray</i>	- HIP mipmapped array
in	<i>level</i>	- Mipmap level

Returns

#hipSuccess, #hipErrorInvalidValue

4.8.2.11 hipHostAlloc()

```
static hipError_t hipHostAlloc (
    void ** ptr,
    size_t size,
    unsigned int flags ) [inline]
```

Allocate device accessible page locked host memory [Deprecated].

Parameters

out	<i>ptr</i>	Pointer to the allocated host pinned memory
in	<i>size</i>	Requested memory size
in	<i>flags</i>	Type of host memory allocation

If size is 0, no memory is allocated, *ptr returns nullptr, and hipSuccess is returned.

Returns

#hipSuccess, #hipErrorOutOfMemory

Parameters

out	<i>ptr</i>	Pointer to the allocated host pinned memory
in	<i>size</i>	Requested memory size
in	<i>flags</i>	Type of host memory allocation

If size is 0, no memory is allocated, *ptr returns nullptr, and hipSuccess is returned.

Returns

#hipSuccess, #hipErrorOutOfMemory

4.8.2.12 hipHostFree()

```
hipError_t hipHostFree (
    void * ptr )
```

Free memory allocated by the hcc hip host memory allocation API This API performs an implicit [hipDeviceSynchronize\(\)](#) call. If pointer is NULL, the hip runtime is initialized and hipSuccess is returned.

Parameters

in	<i>ptr</i>	Pointer to memory to be freed
----	------------	-------------------------------

Returns

#hipSuccess, #hipErrorInvalidValue (if pointer is invalid, including device pointers allocated with hipMalloc)

See also

[hipMalloc](#), [hipMallocPitch](#), [hipFree](#), [hipMallocArray](#), [hipFreeArray](#), [hipMalloc3D](#), [hipMalloc3DArray](#), [hipHostMalloc](#)

4.8.2.13 hipHostGetDevicePointer()

```
hipError_t hipHostGetDevicePointer (
    void ** devPtr,
    void * hstPtr,
    unsigned int flags )
```

Get Device pointer from Host Pointer allocated through hipHostMalloc.

Parameters

out	<i>dstPtr</i>	Device Pointer mapped to passed host pointer
in	<i>hstPtr</i>	Host Pointer allocated through hipHostMalloc
in	<i>flags</i>	Flags to be passed for extension

Returns

#hipSuccess, #hipErrorInvalidValue, #hipErrorOutOfMemory

See also

[hipSetDeviceFlags](#), [hipHostMalloc](#)

4.8.2.14 hipHostGetFlags()

```
hipError_t hipHostGetFlags (
    unsigned int * flagsPtr,
    void * hostPtr )
```

Return flags associated with host pointer.

Parameters

out	<i>flagsPtr</i>	Memory location to store flags
in	<i>hostPtr</i>	Host Pointer allocated through hipHostMalloc

Returns

#hipSuccess, #hipErrorInvalidValue

See also

[hipHostMalloc](#)

4.8.2.15 hipHostMalloc()

```
hipError_t hipHostMalloc (
    void ** ptr,
```

```
size_t size,
unsigned int flags )
```

Allocate device accessible page locked host memory.

Parameters

out	<i>ptr</i>	Pointer to the allocated host pinned memory
in	<i>size</i>	Requested memory size
in	<i>flags</i>	Type of host memory allocation

If size is 0, no memory is allocated, *ptr returns nullptr, and hipSuccess is returned.

Returns

#hipSuccess, #hipErrorOutOfMemory

See also

[hipSetDeviceFlags](#), [hipHostFree](#)

4.8.2.16 hipHostRegister()

```
hipError_t hipHostRegister (
    void * hostPtr,
    size_t sizeBytes,
    unsigned int flags )
```

Register host memory so it can be accessed from the current device.

Parameters

out	<i>hostPtr</i>	Pointer to host memory to be registered.
in	<i>sizeBytes</i>	size of the host memory
in	<i>flags.</i>	See below.

Flags:

- [hipHostRegisterDefault](#) Memory is Mapped and Portable
- [hipHostRegisterPortable](#) Memory is considered registered by all contexts. HIP only supports one context so this is always assumed true.
- [hipHostRegisterMapped](#) Map the allocation into the address space for the current device. The device pointer can be obtained with [hipHostGetDevicePointer](#).

After registering the memory, use [hipHostGetDevicePointer](#) to obtain the mapped device pointer. On many systems, the mapped device pointer will have a different value than the mapped host pointer. Applications must use the device pointer in device code, and the host pointer in device code.

On some systems, registered memory is pinned. On some systems, registered memory may not be actually be pinned but uses OS or hardware facilities to all GPU access to the host memory.

Developers are strongly encouraged to register memory blocks which are aligned to the host cache-line size. (typically 64-bytes but can be obtains from the CPUID instruction).

If registering non-aligned pointers, the application must take care when register pointers from the same cache line on different devices. HIP's coarse-grained synchronization model does not guarantee correct results if different devices write to different parts of the same cache block - typically one of the writes will "win" and overwrite data from the other registered memory region.

Returns

#hipSuccess, #hipErrorOutOfMemory

See also

[hipHostUnregister](#), [hipHostGetFlags](#), [hipHostGetDevicePointer](#)

4.8.2.17 hipHostUnregister()

```
hipError_t hipHostUnregister (
    void * hostPtr )
```

Un-register host pointer.

Parameters

in	<i>hostPtr</i>	Host pointer previously registered with hipHostRegister
----	----------------	---

Returns

Error code

See also

[hipHostRegister](#)

4.8.2.18 hipImportExternalMemory()

```
hipError_t hipImportExternalMemory (
    hipExternalMemory_t * extMem_out,
    const hipExternalMemoryHandleDesc * memHandleDesc )
```

Imports an external memory object.

Parameters

out	<i>extMem_out</i>	Returned handle to an external memory object
in	<i>memHandleDesc</i>	Memory import handle descriptor

Returns

#hipSuccess, #hipErrorInvalidDevice, #hipErrorInvalidValue

See also

4.8.2.19 hipMalloc()

```
hipError_t hipMalloc (
    void ** ptr,
    size_t size )
```

Allocate memory on the default accelerator.

Parameters

out	<i>ptr</i>	Pointer to the allocated memory
in	<i>size</i>	Requested memory size

If size is 0, no memory is allocated, *ptr returns nullptr, and hipSuccess is returned.

Returns

#hipSuccess, #hipErrorOutOfMemory, #hipErrorInvalidValue (bad context, null *ptr)

See also

[hipMallocPitch](#), [hipFree](#), [hipMallocArray](#), [hipFreeArray](#), [hipMalloc3D](#), [hipMalloc3DArray](#), [hipHostFree](#), [hipHostMalloc](#)

4.8.2.20 hipMalloc3DArray()

```
hipError_t hipMalloc3DArray (
    hipArray ** array,
    const struct hipChannelFormatDesc * desc,
    struct hipExtent extent,
    unsigned int flags )
```

Allocate an array on the device.

Parameters

out	<i>array</i>	Pointer to allocated array in device memory
in	<i>desc</i>	Requested channel format
in	<i>extent</i>	Requested array allocation width, height and depth
in	<i>flags</i>	Requested properties of allocated array

Returns

#hipSuccess, #hipErrorOutOfMemory

See also

[hipMalloc](#), [hipMallocPitch](#), [hipFree](#), [hipFreeArray](#), [hipHostMalloc](#), [hipHostFree](#)

4.8.2.21 hipMallocArray()

```
hipError_t hipMallocArray (
    hipArray ** array,
    const hipChannelFormatDesc * desc,
    size_t width,
    size_t height __dparm0,
    unsigned int flags __dparmhipArrayDefault )
```

Allocate an array on the device.

Parameters

out	<i>array</i>	Pointer to allocated array in device memory
in	<i>desc</i>	Requested channel format
in	<i>width</i>	Requested array allocation width

Parameters

in	<i>height</i>	Requested array allocation height
in	<i>flags</i>	Requested properties of allocated array

Returns

#hipSuccess, #hipErrorOutOfMemory

See also

[hipMalloc](#), [hipMallocPitch](#), [hipFree](#), [hipFreeArray](#), [hipHostMalloc](#), [hipHostFree](#)

4.8.2.22 hipMallocHost()

```
static hipError_t hipMallocHost (
    void ** ptr,
    size_t size ) [inline]
```

Allocate pinned host memory [Deprecated].

Parameters

out	<i>ptr</i>	Pointer to the allocated host pinned memory
in	<i>size</i>	Requested memory size

If size is 0, no memory is allocated, *ptr returns nullptr, and hipSuccess is returned.

Returns

#hipSuccess, #hipErrorOutOfMemory

Parameters

out	<i>ptr</i>	Pointer to the allocated host pinned memory
in	<i>size</i>	Requested memory size

If size is 0, no memory is allocated, *ptr returns nullptr, and hipSuccess is returned.

Returns

#hipSuccess, #hipErrorOutOfMemory

4.8.2.23 hipMallocMipmappedArray()

```
hipError_t hipMallocMipmappedArray (
    hipMipmappedArray_t * mipmappedArray,
    const struct hipChannelFormatDesc * desc,
    struct hipExtent extent,
    unsigned int numLevels,
    unsigned int flags __dparm0 )
```

Allocate a mipmapped array on the device.

Parameters

out	<i>mipmappedArray</i>	- Pointer to allocated mipmapped array in device memory
-----	-----------------------	---

Parameters

in	<i>desc</i>	- Requested channel format
in	<i>extent</i>	- Requested allocation size (width field in elements)
in	<i>numLevels</i>	- Number of mipmap levels to allocate
in	<i>flags</i>	- Flags for extensions

Returns

#hipSuccess, #hipErrorInvalidValue, #hipErrorMemoryAllocation

4.8.2.24 hipMallocPitch()

```
hipError_t hipMallocPitch (
    void ** ptr,
    size_t * pitch,
    size_t width,
    size_t height )
```

Allocates at least width (in bytes) * height bytes of linear memory Padding may occur to ensure alignment requirements are met for the given row The change in width size due to padding will be returned in *pitch. Currently the alignment is set to 128 bytes

Parameters

out	<i>ptr</i>	Pointer to the allocated device memory
out	<i>pitch</i>	Pitch for allocation (in bytes)
in	<i>width</i>	Requested pitched allocation width (in bytes)
in	<i>height</i>	Requested pitched allocation height

If size is 0, no memory is allocated, *ptr returns nullptr, and hipSuccess is returned.

Returns

Error code

See also

[hipMalloc](#), [hipFree](#), [hipMallocArray](#), [hipFreeArray](#), [hipHostFree](#), [hipMalloc3D](#), [hipMalloc3DArray](#), [hipHostMalloc](#)

4.8.2.25 hipMemAllocHost()

```
static hipError_t hipMemAllocHost (
    void ** ptr,
    size_t size ) [inline]
```

Allocate pinned host memory [Deprecated].

Parameters

out	<i>ptr</i>	Pointer to the allocated host pinned memory
in	<i>size</i>	Requested memory size

If size is 0, no memory is allocated, *ptr returns nullptr, and hipSuccess is returned.

Returns

#hipSuccess, #hipErrorOutOfMemory

Parameters

out	<i>ptr</i>	Pointer to the allocated host pinned memory
in	<i>size</i>	Requested memory size

If size is 0, no memory is allocated, *ptr returns nullptr, and hipSuccess is returned.

Returns

#hipSuccess, #hipErrorOutOfMemory

4.8.2.26 hipMemAllocPitch()

```
hipError_t hipMemAllocPitch (
    hipDeviceptr_t * dptr,
    size_t * pitch,
    size_t widthInBytes,
    size_t height,
    unsigned int elementSizeBytes )
```

Allocates at least width (in bytes) * height bytes of linear memory. Padding may occur to ensure alignment requirements are met for the given row. The change in width size due to padding will be returned in *pitch. Currently the alignment is set to 128 bytes.

Parameters

out	<i>dptr</i>	Pointer to the allocated device memory
out	<i>pitch</i>	Pitch for allocation (in bytes)
in	<i>width</i>	Requested pitched allocation width (in bytes)
in	<i>height</i>	Requested pitched allocation height

If size is 0, no memory is allocated, *ptr* returns nullptr, and *hipSuccess* is returned. The intended usage of *pitch* is as a separate parameter of the allocation, used to compute addresses within the 2D array. Given the row and column of an array element of type *T*, the address is computed as: $T \text{ pElement} = (T*)((\text{char}*)\text{BaseAddress} + \text{Row} * \text{Pitch}) + \text{Column};$

Returns

Error code

See also

[hipMalloc](#), [hipFree](#), [hipMallocArray](#), [hipFreeArray](#), [hipHostFree](#), [hipMalloc3D](#), [hipMalloc3DArray](#), [hipHostMalloc](#)

4.8.2.27 hipMemcpy()

```
hipError_t hipMemcpy (
    void * dst,
    const void * src,
    size_t sizeBytes,
    hipMemcpyKind kind )
```

Copy data from src to dst.

It supports memory from host to device, device to host, device to device and host to host. The src and dst must not overlap.

For `hipMemcpy`, the copy is always performed by the current device (set by `hipSetDevice`). For multi-gpu or peer-to-peer configurations, it is recommended to set the current device to the device where the src data is physically located. For optimal peer-to-peer copies, the copy device must be able to access the src and dst pointers (by calling `hipDeviceEnablePeerAccess` with copy agent as the current device and src/dest as the peerDevice argument). If this is not done, the `hipMemcpy` will still work, but will perform the copy using a staging buffer on the host. Calling `hipMemcpy` with dst and src pointers that do not match the `hipMemcpyKind` results in undefined behavior.

Parameters

out	<i>dst</i>	Data being copy to
in	<i>src</i>	Data being copy from
in	<i>sizeBytes</i>	Data size in bytes
in	<i>copyType</i>	Memory copy type

Returns

`#hipSuccess`, `#hipErrorInvalidValue`, `#hipErrorMemoryFree`, `#hipErrorUnknown`

See also

`hipArrayCreate`, `hipArrayDestroy`, `hipArrayGetDescriptor`, `hipMemAlloc`, [hipMemAllocHost](#), [hipMemAllocPitch](#), [hipMemcpy2D](#), [hipMemcpy2DAsync](#), `hipMemcpy2DUnaligned`, `hipMemcpyAtoA`, `hipMemcpyAtoD`, [hipMemcpyAtoH](#), `hipMemcpyAtoHAsync`, `hipMemcpyDtoA`, [hipMemcpyDtoD](#), [hipMemcpyDtoDAsync](#), [hipMemcpyDtoH](#), [hipMemcpyDtoHAsync](#), [hipMemcpyHtoA](#), `hipMemcpyHtoAAsync`, [hipMemcpyHtoDAsync](#), `hipMemFree`, `hipMemFreeHost`, [hipMemGetAddressRange](#), [hipMemGetInfo](#), `hipMemHostAlloc`, `hipMemHostGetDevicePointer`

4.8.2.28 hipMemcpy2D()

```
hipError_t hipMemcpy2D (
    void * dst,
    size_t dpitch,
    const void * src,
    size_t spitch,
    size_t width,
    size_t height,
    hipMemcpyKind kind )
```

Copies data between host and device.

Parameters

in	<i>dst</i>	Destination memory address
in	<i>dpitch</i>	Pitch of destination memory
in	<i>src</i>	Source memory address
in	<i>spitch</i>	Pitch of source memory
in	<i>width</i>	Width of matrix transfer (columns in bytes)
in	<i>height</i>	Height of matrix transfer (rows)
in	<i>kind</i>	Type of transfer

Returns

#hipSuccess, #hipErrorInvalidValue, #hipErrorInvalidPitchValue, #hipErrorInvalidDevicePointer, #hipErrorInvalidMemcpyDirection

See also

[hipMemcpy](#), [hipMemcpyToArray](#), [hipMemcpy2DToArray](#), [hipMemcpyFromArray](#), [hipMemcpyToSymbol](#), [hipMemcpyAsync](#)

4.8.2.29 hipMemcpy2DAsync()

```
hipError_t hipMemcpy2DAsync (
    void * dst,
    size_t dpitch,
    const void * src,
    size_t spitch,
    size_t width,
    size_t height,
    hipMemcpyKind kind,
    hipStream_t stream __dparm0 )
```

Copies data between host and device.

Parameters

in	<i>dst</i>	Destination memory address
in	<i>dpitch</i>	Pitch of destination memory
in	<i>src</i>	Source memory address
in	<i>spitch</i>	Pitch of source memory
in	<i>width</i>	Width of matrix transfer (columns in bytes)
in	<i>height</i>	Height of matrix transfer (rows)
in	<i>kind</i>	Type of transfer
in	<i>stream</i>	Stream to use

Returns

#hipSuccess, #hipErrorInvalidValue, #hipErrorInvalidPitchValue, #hipErrorInvalidDevicePointer, #hipErrorInvalidMemcpyDirection

See also

[hipMemcpy](#), [hipMemcpyToArray](#), [hipMemcpy2DToArray](#), [hipMemcpyFromArray](#), [hipMemcpyToSymbol](#), [hipMemcpyAsync](#)

4.8.2.30 hipMemcpy2DFromArray()

```
hipError_t hipMemcpy2DFromArray (
    void * dst,
    size_t dpitch,
    hipArray_const_t src,
    size_t wOffset,
    size_t hOffset,
    size_t width,
    size_t height,
    hipMemcpyKind kind )
```

Copies data between host and device.

Parameters

in	<i>dst</i>	Destination memory address
in	<i>dpitch</i>	Pitch of destination memory
in	<i>src</i>	Source memory address
in	<i>wOffset</i>	Source starting X offset
in	<i>hOffset</i>	Source starting Y offset
in	<i>width</i>	Width of matrix transfer (columns in bytes)
in	<i>height</i>	Height of matrix transfer (rows)
in	<i>kind</i>	Type of transfer

Returns

#hipSuccess, #hipErrorInvalidValue, #hipErrorInvalidPitchValue, #hipErrorInvalidDevicePointer, #hipErrorInvalidMemcpyDirection

See also

[hipMemcpy](#), [hipMemcpy2DToArray](#), [hipMemcpy2D](#), [hipMemcpyFromArray](#), [hipMemcpyToSymbol](#), [hipMemcpyAsync](#)

4.8.2.31 hipMemcpy2DFromArrayAsync()

```
hipError_t hipMemcpy2DFromArrayAsync (
    void * dst,
    size_t dpitch,
    hipArray_const_t src,
    size_t wOffset,
    size_t hOffset,
    size_t width,
    size_t height,
    hipMemcpyKind kind,
    hipStream_t stream __dparm0 )
```

Copies data between host and device asynchronously.

Parameters

in	<i>dst</i>	Destination memory address
in	<i>dpitch</i>	Pitch of destination memory
in	<i>src</i>	Source memory address
in	<i>wOffset</i>	Source starting X offset
in	<i>hOffset</i>	Source starting Y offset
in	<i>width</i>	Width of matrix transfer (columns in bytes)
in	<i>height</i>	Height of matrix transfer (rows)
in	<i>kind</i>	Type of transfer
in	<i>stream</i>	Accelerator view which the copy is being enqueued

Returns

#hipSuccess, #hipErrorInvalidValue, #hipErrorInvalidPitchValue, #hipErrorInvalidDevicePointer, #hipErrorInvalidMemcpyDirection

See also

[hipMemcpy](#), [hipMemcpy2DToArray](#), [hipMemcpy2D](#), [hipMemcpyFromArray](#), [hipMemcpyToSymbol](#), [hipMemcpyAsync](#)

4.8.2.32 hipMemcpy2DToArray()

```
hipError_t hipMemcpy2DToArray (
    hipArray * dst,
    size_t wOffset,
    size_t hOffset,
    const void * src,
    size_t spitch,
    size_t width,
    size_t height,
    hipMemcpyKind kind )
```

Copies data between host and device.

Parameters

in	<i>dst</i>	Destination memory address
in	<i>wOffset</i>	Destination starting X offset
in	<i>hOffset</i>	Destination starting Y offset
in	<i>src</i>	Source memory address
in	<i>spitch</i>	Pitch of source memory
in	<i>width</i>	Width of matrix transfer (columns in bytes)
in	<i>height</i>	Height of matrix transfer (rows)
in	<i>kind</i>	Type of transfer

Returns

#hipSuccess, #hipErrorInvalidValue, #hipErrorInvalidPitchValue, #hipErrorInvalidDevicePointer, #hipErrorInvalidMemcpyDirection

See also

[hipMemcpy](#), [hipMemcpyToArray](#), [hipMemcpy2D](#), [hipMemcpyFromArray](#), [hipMemcpyToSymbol](#), [hipMemcpyAsync](#)

4.8.2.33 hipMemcpy2DToArrayAsync()

```
hipError_t hipMemcpy2DToArrayAsync (
    hipArray * dst,
    size_t wOffset,
    size_t hOffset,
    const void * src,
    size_t spitch,
    size_t width,
    size_t height,
    hipMemcpyKind kind,
    hipStream_t stream __dparm0 )
```

Copies data between host and device.

Parameters

in	<i>dst</i>	Destination memory address
----	------------	----------------------------

Parameters

in	<i>wOffset</i>	Destination starting X offset
in	<i>hOffset</i>	Destination starting Y offset
in	<i>src</i>	Source memory address
in	<i>spitch</i>	Pitch of source memory
in	<i>width</i>	Width of matrix transfer (columns in bytes)
in	<i>height</i>	Height of matrix transfer (rows)
in	<i>kind</i>	Type of transfer
in	<i>stream</i>	Accelerator view which the copy is being enqueued

Returns

#hipSuccess, #hipErrorInvalidValue, #hipErrorInvalidPitchValue, #hipErrorInvalidDevicePointer, #hipErrorInvalidMemcpyDirection

See also

[hipMemcpy](#), [hipMemcpyToArray](#), [hipMemcpy2D](#), [hipMemcpyFromArray](#), [hipMemcpyToSymbol](#), [hipMemcpyAsync](#)

4.8.2.34 hipMemcpy3D()

```
hipError_t hipMemcpy3D (
    const struct hipMemcpy3DParms * p )
```

Copies data between host and device.

Parameters

in	<i>p</i>	3D memory copy parameters
----	----------	---------------------------

Returns

#hipSuccess, #hipErrorInvalidValue, #hipErrorInvalidPitchValue, #hipErrorInvalidDevicePointer, #hipErrorInvalidMemcpyDirection

See also

[hipMemcpy](#), [hipMemcpy2DToArray](#), [hipMemcpy2D](#), [hipMemcpyFromArray](#), [hipMemcpyToSymbol](#), [hipMemcpyAsync](#)

4.8.2.35 hipMemcpy3DAsync()

```
hipError_t hipMemcpy3DAsync (
    const struct hipMemcpy3DParms * p,
    hipStream_t stream __dparm0 )
```

Copies data between host and device asynchronously.

Parameters

in	<i>p</i>	3D memory copy parameters
in	<i>stream</i>	Stream to use

Returns

#hipSuccess, #hipErrorInvalidValue, #hipErrorInvalidPitchValue, #hipErrorInvalidDevicePointer, #hipErrorInvalidMemcpyDirection

See also

[hipMemcpy](#), [hipMemcpy2DToArray](#), [hipMemcpy2D](#), [hipMemcpyFromArray](#), [hipMemcpyToSymbol](#), [hipMemcpyAsync](#)

4.8.2.36 hipMemcpyAsync()

```
hipError_t hipMemcpyAsync (
    void * dst,
    const void * src,
    size_t sizeBytes,
    hipMemcpyKind kind,
    hipStream_t stream __dparm0 )
```

Copy data from src to dst asynchronously.

Warning

If host or dest are not pinned, the memory copy will be performed synchronously. For best performance, use `hipHostMalloc` to allocate host memory that is transferred asynchronously.

on HCC `hipMemcpyAsync` does not support overlapped H2D and D2H copies. For `hipMemcpy`, the copy is always performed by the device associated with the specified stream.

For multi-gpu or peer-to-peer configurations, it is recommended to use a stream which is attached to the device where the src data is physically located. For optimal peer-to-peer copies, the copy device must be able to access the src and dst pointers (by calling `hipDeviceEnablePeerAccess` with copy agent as the current device and src/dest as the peerDevice argument. if this is not done, the `hipMemcpy` will still work, but will perform the copy using a staging buffer on the host.

Parameters

out	<i>dst</i>	Data being copy to
in	<i>src</i>	Data being copy from
in	<i>sizeBytes</i>	Data size in bytes
in	<i>accelerator_view</i>	Accelerator view which the copy is being enqueued

Returns

#hipSuccess, #hipErrorInvalidValue, #hipErrorMemoryFree, #hipErrorUnknown

See also

[hipMemcpy](#), [hipMemcpy2D](#), [hipMemcpyToArray](#), [hipMemcpy2DToArray](#), [hipMemcpyFromArray](#), [hipMemcpy2DFromArray](#), [hipMemcpyArrayToArray](#), [hipMemcpy2DArrayToArray](#), [hipMemcpyToSymbol](#), [hipMemcpyFromSymbol](#), [hipMemcpy2DAsync](#), [hipMemcpyToArrayAsync](#), [hipMemcpy2DToArrayAsync](#), [hipMemcpyFromArrayAsync](#), [hipMemcpy2DFromArrayAsync](#), [hipMemcpyToSymbolAsync](#), [hipMemcpyFromSymbolAsync](#)

4.8.2.37 hipMemcpyAtoH()

```
hipError_t hipMemcpyAtoH (
    void * dst,
    hipArray * srcArray,
```

```
size_t srcOffset,
size_t count )
```

Copies data between host and device.

Parameters

in	<i>dst</i>	Destination memory address
in	<i>srcArray</i>	Source array
in	<i>srcOffset</i>	Offset in bytes of source array
in	<i>count</i>	Size of memory copy in bytes

Returns

#hipSuccess, #hipErrorInvalidValue, #hipErrorInvalidPitchValue, #hipErrorInvalidDevicePointer, #hipErrorInvalidMemcpyDirection

See also

[hipMemcpy](#), [hipMemcpy2DToArray](#), [hipMemcpy2D](#), [hipMemcpyFromArray](#), [hipMemcpyToSymbol](#), [hipMemcpyAsync](#)

4.8.2.38 hipMemcpyDtoD()

```
hipError_t hipMemcpyDtoD (
    hipDeviceptr_t dst,
    hipDeviceptr_t src,
    size_t sizeBytes )
```

Copy data from Device to Device.

Parameters

out	<i>dst</i>	Data being copy to
in	<i>src</i>	Data being copy from
in	<i>sizeBytes</i>	Data size in bytes

Returns

#hipSuccess, #hipErrorDeinitialized, #hipErrorNotInitialized, #hipErrorInvalidContext, #hipErrorInvalidValue

See also

[hipArrayCreate](#), [hipArrayDestroy](#), [hipArrayGetDescriptor](#), [hipMemAlloc](#), [hipMemAllocHost](#), [hipMemAllocPitch](#), [hipMemcpy2D](#), [hipMemcpy2DAsync](#), [hipMemcpy2DUnaligned](#), [hipMemcpyAtoA](#), [hipMemcpyAtoD](#), [hipMemcpyAtoH](#), [hipMemcpyAtoHAsync](#), [hipMemcpyDtoA](#), [hipMemcpyDtoD](#), [hipMemcpyDtoDAsync](#), [hipMemcpyDtoH](#), [hipMemcpyDtoHAsync](#), [hipMemcpyHtoA](#), [hipMemcpyHtoAAsync](#), [hipMemcpyHtoDAsync](#), [hipMemFree](#), [hipMemFreeHost](#), [hipMemGetAddressRange](#), [hipMemGetInfo](#), [hipMemHostAlloc](#), [hipMemHostGetDevicePointer](#)

4.8.2.39 hipMemcpyDtoDAsync()

```
hipError_t hipMemcpyDtoDAsync (
    hipDeviceptr_t dst,
    hipDeviceptr_t src,
    size_t sizeBytes,
    hipStream_t stream )
```

Copy data from Device to Device asynchronously.

Parameters

out	<i>dst</i>	Data being copy to
in	<i>src</i>	Data being copy from
in	<i>sizeBytes</i>	Data size in bytes

Returns

#hipSuccess, #hipErrorDeinitialized, #hipErrorNotInitialized, #hipErrorInvalidContext, #hipErrorInvalidValue

See also

hipArrayCreate, hipArrayDestroy, hipArrayGetDescriptor, hipMemAlloc, [hipMemAllocHost](#), [hipMemAllocPitch](#), [hipMemcpy2D](#), [hipMemcpy2DAsync](#), hipMemcpy2DUnaligned, hipMemcpyAtoA, hipMemcpyAtoD, [hipMemcpyAtoH](#), hipMemcpyAtoHAsync, hipMemcpyDtoA, [hipMemcpyDtoD](#), [hipMemcpyDtoDAsync](#), [hipMemcpyDtoH](#), [hipMemcpyDtoHAsync](#), [hipMemcpyHtoA](#), hipMemcpyHtoAAsync, [hipMemcpyHtoDAsync](#), hipMemFree, hipMemFreeHost, [hipMemGetAddressRange](#), [hipMemGetInfo](#), hipMemHostAlloc, hipMem↔HostGetDevicePointer

4.8.2.40 hipMemcpyDtoH()

```
hipError_t hipMemcpyDtoH (
    void * dst,
    hipDeviceptr_t src,
    size_t sizeBytes )
```

Copy data from Device to Host.

Parameters

out	<i>dst</i>	Data being copy to
in	<i>src</i>	Data being copy from
in	<i>sizeBytes</i>	Data size in bytes

Returns

#hipSuccess, #hipErrorDeinitialized, #hipErrorNotInitialized, #hipErrorInvalidContext, #hipErrorInvalidValue

See also

hipArrayCreate, hipArrayDestroy, hipArrayGetDescriptor, hipMemAlloc, [hipMemAllocHost](#), [hipMemAllocPitch](#), [hipMemcpy2D](#), [hipMemcpy2DAsync](#), hipMemcpy2DUnaligned, hipMemcpyAtoA, hipMemcpyAtoD, [hipMemcpyAtoH](#), hipMemcpyAtoHAsync, hipMemcpyDtoA, [hipMemcpyDtoD](#), [hipMemcpyDtoDAsync](#), [hipMemcpyDtoH](#), [hipMemcpyDtoHAsync](#), [hipMemcpyHtoA](#), hipMemcpyHtoAAsync, [hipMemcpyHtoDAsync](#), hipMemFree, hipMemFreeHost, [hipMemGetAddressRange](#), [hipMemGetInfo](#), hipMemHostAlloc, hipMem↔HostGetDevicePointer

4.8.2.41 hipMemcpyDtoHAsync()

```
hipError_t hipMemcpyDtoHAsync (
    void * dst,
    hipDeviceptr_t src,
    size_t sizeBytes,
    hipStream_t stream )
```

Copy data from Device to Host asynchronously.

Parameters

out	<i>dst</i>	Data being copy to
in	<i>src</i>	Data being copy from
in	<i>sizeBytes</i>	Data size in bytes

Returns

#hipSuccess, #hipErrorDeinitialized, #hipErrorNotInitialized, #hipErrorInvalidContext, #hipErrorInvalidValue

See also

hipArrayCreate, hipArrayDestroy, hipArrayGetDescriptor, hipMemAlloc, [hipMemAllocHost](#), [hipMemAllocPitch](#), [hipMemcpy2D](#), [hipMemcpy2DAsync](#), hipMemcpy2DUnaligned, hipMemcpyAtoA, hipMemcpyAtoD, [hipMemcpyAtoH](#), hipMemcpyAtoHAsync, hipMemcpyDtoA, [hipMemcpyDtoD](#), [hipMemcpyDtoDAsync](#), [hipMemcpyDtoH](#), [hipMemcpyDtoHAsync](#), [hipMemcpyHtoA](#), hipMemcpyHtoAAsync, [hipMemcpyHtoDAsync](#), hipMemFree, hipMemFreeHost, [hipMemGetAddressRange](#), [hipMemGetInfo](#), hipMemHostAlloc, hipMem↔HostGetDevicePointer

4.8.2.42 hipMemcpyFromArray()

```
hipError_t hipMemcpyFromArray (
    void * dst,
    hipArray_const_t srcArray,
    size_t wOffset,
    size_t hOffset,
    size_t count,
    hipMemcpyKind kind )
```

Copies data between host and device.

Parameters

in	<i>dst</i>	Destination memory address
in	<i>srcArray</i>	Source memory address
in	<i>wOffset</i>	Source starting X offset
in	<i>hOffset</i>	Source starting Y offset
in	<i>count</i>	Size in bytes to copy
in	<i>kind</i>	Type of transfer

Returns

#hipSuccess, #hipErrorInvalidValue, #hipErrorInvalidPitchValue, #hipErrorInvalidDevicePointer, #hipError↔InvalidMemcpyDirection

See also

[hipMemcpy](#), [hipMemcpy2DToArray](#), [hipMemcpy2D](#), [hipMemcpyFromArray](#), [hipMemcpyToSymbol](#), [hipMemcpyAsync](#)

4.8.2.43 hipMemcpyHtoA()

```
hipError_t hipMemcpyHtoA (
    hipArray * dstArray,
    size_t dstOffset,
```



```
const void * srcHost,
size_t count )
```

Copies data between host and device.

Parameters

in	<i>dstArray</i>	Destination memory address
in	<i>dstOffset</i>	Offset in bytes of destination array
in	<i>srcHost</i>	Source host pointer
in	<i>count</i>	Size of memory copy in bytes

Returns

#hipSuccess, #hipErrorInvalidValue, #hipErrorInvalidPitchValue, #hipErrorInvalidDevicePointer, #hipErrorInvalidMemcpyDirection

See also

[hipMemcpy](#), [hipMemcpy2DToArray](#), [hipMemcpy2D](#), [hipMemcpyFromArray](#), [hipMemcpyToSymbol](#), [hipMemcpyAsync](#)

4.8.2.44 hipMemcpyHtoD()

```
hipError_t hipMemcpyHtoD (
    hipDeviceptr_t dst,
    void * src,
    size_t sizeBytes )
```

Copy data from Host to Device.

Parameters

out	<i>dst</i>	Data being copy to
in	<i>src</i>	Data being copy from
in	<i>sizeBytes</i>	Data size in bytes

Returns

#hipSuccess, #hipErrorDeinitialized, #hipErrorNotInitialized, #hipErrorInvalidContext, #hipErrorInvalidValue

See also

[hipArrayCreate](#), [hipArrayDestroy](#), [hipArrayGetDescriptor](#), [hipMemAlloc](#), [hipMemAllocHost](#), [hipMemAllocPitch](#), [hipMemcpy2D](#), [hipMemcpy2DAsync](#), [hipMemcpy2DUnaligned](#), [hipMemcpyAtoA](#), [hipMemcpyAtoD](#), [hipMemcpyAtoH](#), [hipMemcpyAtoHAsync](#), [hipMemcpyDtoA](#), [hipMemcpyDtoD](#), [hipMemcpyDtoDAsync](#), [hipMemcpyDtoH](#), [hipMemcpyDtoHAsync](#), [hipMemcpyHtoA](#), [hipMemcpyHtoAAsync](#), [hipMemcpyHtoDAsync](#), [hipMemFree](#), [hipMemFreeHost](#), [hipMemGetAddressRange](#), [hipMemGetInfo](#), [hipMemHostAlloc](#), [hipMemHostGetDevicePointer](#)

4.8.2.45 hipMemcpyHtoDAsync()

```
hipError_t hipMemcpyHtoDAsync (
    hipDeviceptr_t dst,
    void * src,
    size_t sizeBytes,
    hipStream_t stream )
```

Copy data from Host to Device asynchronously.

Parameters

out	<i>dst</i>	Data being copy to
in	<i>src</i>	Data being copy from
in	<i>sizeBytes</i>	Data size in bytes

Returns

#hipSuccess, #hipErrorDeinitialized, #hipErrorNotInitialized, #hipErrorInvalidContext, #hipErrorInvalidValue

See also

hipArrayCreate, hipArrayDestroy, hipArrayGetDescriptor, hipMemAlloc, [hipMemAllocHost](#), [hipMemAllocPitch](#), [hipMemcpy2D](#), [hipMemcpy2DAsync](#), hipMemcpy2DUnaligned, hipMemcpyAtoA, hipMemcpyAtoD, [hipMemcpyAtoH](#), hipMemcpyAtoHAsync, hipMemcpyDtoA, [hipMemcpyDtoD](#), [hipMemcpyDtoDAsync](#), [hipMemcpyDtoH](#), [hipMemcpyDtoHAsync](#), [hipMemcpyHtoA](#), hipMemcpyHtoAAsync, [hipMemcpyHtoDAsync](#), hipMemFree, hipMemFreeHost, [hipMemGetAddressRange](#), [hipMemGetInfo](#), hipMemHostAlloc, hipMem↵
HostGetDevicePointer

4.8.2.46 hipMemcpyParam2D()

```
hipError_t hipMemcpyParam2D (
    const hip\_Memcpy2D * pCopy )
```

Copies memory for 2D arrays.

Parameters

in	<i>pCopy</i>	Parameters for the memory copy
----	--------------	--------------------------------

Returns

#hipSuccess, #hipErrorInvalidValue, #hipErrorInvalidPitchValue, #hipErrorInvalidDevicePointer, #hipError↵
InvalidMemcpyDirection

See also

[hipMemcpy](#), [hipMemcpy2D](#), [hipMemcpyToArray](#), [hipMemcpy2DToArray](#), [hipMemcpyFromArray](#), hipMemcpy↵
ToSymbol, [hipMemcpyAsync](#)

4.8.2.47 hipMemcpyParam2DAsync()

```
hipError_t hipMemcpyParam2DAsync (
    const hip\_Memcpy2D * pCopy,
    hipStream_t stream __dparm0 )
```

Copies memory for 2D arrays.

Parameters

in	<i>pCopy</i>	Parameters for the memory copy
in	<i>stream</i>	Stream to use

Returns

#hipSuccess, #hipErrorInvalidValue, #hipErrorInvalidPitchValue, #hipErrorInvalidDevicePointer, #hipErrorInvalidMemcpyDirection

See also

[hipMemcpy](#), [hipMemcpy2D](#), [hipMemcpyToArray](#), [hipMemcpy2DToArray](#), [hipMemcpyFromArray](#), [hipMemcpyToSymbol](#), [hipMemcpyAsync](#)

4.8.2.48 hipMemcpyToArray()

```
hipError_t hipMemcpyToArray (
    hipArray * dst,
    size_t wOffset,
    size_t hOffset,
    const void * src,
    size_t count,
    hipMemcpyKind kind )
```

Copies data between host and device.

Parameters

in	<i>dst</i>	Destination memory address
in	<i>wOffset</i>	Destination starting X offset
in	<i>hOffset</i>	Destination starting Y offset
in	<i>src</i>	Source memory address
in	<i>count</i>	size in bytes to copy
in	<i>kind</i>	Type of transfer

Returns

#hipSuccess, #hipErrorInvalidValue, #hipErrorInvalidPitchValue, #hipErrorInvalidDevicePointer, #hipErrorInvalidMemcpyDirection

See also

[hipMemcpy](#), [hipMemcpy2DToArray](#), [hipMemcpy2D](#), [hipMemcpyFromArray](#), [hipMemcpyToSymbol](#), [hipMemcpyAsync](#)

4.8.2.49 hipMemGetInfo()

```
hipError_t hipMemGetInfo (
    size_t * free,
    size_t * total )
```

Query memory info. Return snapshot of free memory, and total allocatable memory on the device.

Returns in *free a snapshot of the current free memory.

Returns

#hipSuccess, #hipErrorInvalidDevice, #hipErrorInvalidValue

Warning

On HCC, the free memory only accounts for memory allocated by this process and may be optimistic.

4.8.2.50 hipMemset()

```
hipError_t hipMemset (
    void * dst,
    int value,
    size_t sizeBytes )
```

Fills the first sizeBytes bytes of the memory area pointed to by dest with the constant byte value value.

Parameters

out	<i>dst</i>	Data being filled
in	<i>constant</i>	value to be set
in	<i>sizeBytes</i>	Data size in bytes

Returns

#hipSuccess, #hipErrorInvalidValue, #hipErrorNotInitialized

4.8.2.51 hipMemset2D()

```
hipError_t hipMemset2D (
    void * dst,
    size_t pitch,
    int value,
    size_t width,
    size_t height )
```

Fills the memory area pointed to by dst with the constant value.

Parameters

out	<i>dst</i>	Pointer to device memory
in	<i>pitch</i>	- data size in bytes
in	<i>value</i>	- constant value to be set
in	<i>width</i>	
in	<i>height</i>	

Returns

#hipSuccess, #hipErrorInvalidValue, #hipErrorMemoryFree

4.8.2.52 hipMemset2DAsync()

```
hipError_t hipMemset2DAsync (
    void * dst,
    size_t pitch,
    int value,
    size_t width,
    size_t height,
    hipStream_t stream __dparm0 )
```

Fills asynchronously the memory area pointed to by dst with the constant value.

Parameters

in	<i>dst</i>	Pointer to device memory
----	------------	--------------------------

Parameters

in	<i>pitch</i>	- data size in bytes
in	<i>value</i>	- constant value to be set
in	<i>width</i>	
in	<i>height</i>	
in	<i>stream</i>	

Returns

#hipSuccess, #hipErrorInvalidValue, #hipErrorMemoryFree

4.8.2.53 hipMemset3D()

```
hipError_t hipMemset3D (
    hipPitchedPtr pitchedDevPtr,
    int value,
    hipExtent extent )
```

Fills synchronously the memory area pointed to by pitchedDevPtr with the constant value.

Parameters

in	<i>pitchedDevPtr</i>	
in	<i>value</i>	- constant value to be set
in	<i>extent</i>	

Returns

#hipSuccess, #hipErrorInvalidValue, #hipErrorMemoryFree

4.8.2.54 hipMemset3DAsync()

```
hipError_t hipMemset3DAsync (
    hipPitchedPtr pitchedDevPtr,
    int value,
    hipExtent extent,
    hipStream_t stream __dparm0 )
```

Fills asynchronously the memory area pointed to by pitchedDevPtr with the constant value.

Parameters

in	<i>pitchedDevPtr</i>	
in	<i>value</i>	- constant value to be set
in	<i>extent</i>	
in	<i>stream</i>	

Returns

#hipSuccess, #hipErrorInvalidValue, #hipErrorMemoryFree

4.8.2.55 hipMemsetAsync()

```
hipError_t hipMemsetAsync (
    void * dst,
    int value,
    size_t sizeBytes,
    hipStream_t stream __dparm0 )
```

Fills the first sizeBytes bytes of the memory area pointed to by dev with the constant byte value value. [hipMemsetAsync\(\)](#) is asynchronous with respect to the host, so the call may return before the memset is complete. The operation can optionally be associated to a stream by passing a non-zero stream argument. If stream is non-zero, the operation may overlap with operations in other streams.

Parameters

out	<i>dst</i>	Pointer to device memory
in	<i>value</i>	- Value to set for each byte of specified memory
in	<i>sizeBytes</i>	- Size in bytes to set
in	<i>stream</i>	- Stream identifier

Returns

#hipSuccess, #hipErrorInvalidValue, #hipErrorMemoryFree

4.8.2.56 hipMemsetD16()

```
hipError_t hipMemsetD16 (
    hipDeviceptr_t dest,
    unsigned short value,
    size_t count )
```

Fills the first sizeBytes bytes of the memory area pointed to by dest with the constant short value value.

Parameters

out	<i>dst</i>	Data ptr to be filled
in	<i>constant</i>	value to be set
in	<i>number</i>	of values to be set

Returns

#hipSuccess, #hipErrorInvalidValue, #hipErrorNotInitialized

4.8.2.57 hipMemsetD16Async()

```
hipError_t hipMemsetD16Async (
    hipDeviceptr_t dest,
    unsigned short value,
    size_t count,
    hipStream_t stream __dparm0 )
```

Fills the first sizeBytes bytes of the memory area pointed to by dest with the constant short value value.

[hipMemsetD16Async\(\)](#) is asynchronous with respect to the host, so the call may return before the memset is complete. The operation can optionally be associated to a stream by passing a non-zero stream argument. If stream is non-zero, the operation may overlap with operations in other streams.

Parameters

out	<i>dst</i>	Data ptr to be filled
in	<i>constant</i>	value to be set
in	<i>number</i>	of values to be set
in	<i>stream</i>	- Stream identifier

Returns

#hipSuccess, #hipErrorInvalidValue, #hipErrorNotInitialized

4.8.2.58 hipMemsetD32()

```
hipError_t hipMemsetD32 (
    hipDeviceptr_t dest,
    int value,
    size_t count )
```

Fills the memory area pointed to by dest with the constant integer value for specified number of times.

Parameters

out	<i>dst</i>	Data being filled
in	<i>constant</i>	value to be set
in	<i>number</i>	of values to be set

Returns

#hipSuccess, #hipErrorInvalidValue, #hipErrorNotInitialized

4.8.2.59 hipMemsetD32Async()

```
hipError_t hipMemsetD32Async (
    hipDeviceptr_t dst,
    int value,
    size_t count,
    hipStream_t stream __dparm0 )
```

Fills the memory area pointed to by dev with the constant integer value for specified number of times.

[hipMemsetD32Async\(\)](#) is asynchronous with respect to the host, so the call may return before the memset is complete. The operation can optionally be associated to a stream by passing a non-zero stream argument. If stream is non-zero, the operation may overlap with operations in other streams.

Parameters

out	<i>dst</i>	Pointer to device memory
in	<i>value</i>	- Value to set for each byte of specified memory
in	<i>count</i>	- number of values to be set
in	<i>stream</i>	- Stream identifier

Returns

#hipSuccess, #hipErrorInvalidValue, #hipErrorMemoryFree

4.8.2.60 hipMemsetD8()

```
hipError_t hipMemsetD8 (
    hipDeviceptr_t dest,
    unsigned char value,
    size_t count )
```

Fills the first sizeBytes bytes of the memory area pointed to by dest with the constant byte value value.

Parameters

out	<i>dst</i>	Data ptr to be filled
in	<i>constant</i>	value to be set
in	<i>number</i>	of values to be set

Returns

#hipSuccess, #hipErrorInvalidValue, #hipErrorNotInitialized

4.8.2.61 hipMemsetD8Async()

```
hipError_t hipMemsetD8Async (
    hipDeviceptr_t dest,
    unsigned char value,
    size_t count,
    hipStream_t stream __dparm0 )
```

Fills the first sizeBytes bytes of the memory area pointed to by dest with the constant byte value value.

[hipMemsetD8Async\(\)](#) is asynchronous with respect to the host, so the call may return before the memset is complete. The operation can optionally be associated to a stream by passing a non-zero stream argument. If stream is non-zero, the operation may overlap with operations in other streams.

Parameters

out	<i>dst</i>	Data ptr to be filled
in	<i>constant</i>	value to be set
in	<i>number</i>	of values to be set
in	<i>stream</i>	- Stream identifier

Returns

#hipSuccess, #hipErrorInvalidValue, #hipErrorNotInitialized

4.8.2.62 hipPointerGetAttributes()

```
hipError_t hipPointerGetAttributes (
    hipPointerAttribute_t * attributes,
    const void * ptr )
```

Return attributes for the specified pointer.

Parameters

out	<i>attributes</i>	for the specified pointer
in	<i>pointer</i>	to get attributes for

Returns

#hipSuccess, #hipErrorInvalidDevice, #hipErrorInvalidValue

See also

[hipGetDeviceCount](#), [hipGetDevice](#), [hipSetDevice](#), [hipChooseDevice](#)

4.9 PeerToPeer Device Memory Access

Macros

- `#define USE_PEER_NON_UNIFIED 1`
- `#define USE_PEER_NON_UNIFIED 1`

Functions

- `hipError_t hipDeviceCanAccessPeer` (int *canAccessPeer, int deviceId, int peerDeviceId)
Determine if a device can access a peer's memory.
- `hipError_t hipDeviceEnablePeerAccess` (int peerDeviceId, unsigned int flags)
Enable direct access from current device's virtual address space to memory allocations physically located on a peer device.
- `hipError_t hipDeviceDisablePeerAccess` (int peerDeviceId)
Disable direct access from current device's virtual address space to memory allocations physically located on a peer device.
- `hipError_t hipMemGetAddressRange` (hipDeviceptr_t *pbase, size_t *psize, hipDeviceptr_t dptr)
Get information on memory allocations.
- `hipError_t hipMemcpyPeer` (void *dst, int dstDeviceId, const void *src, int srcDeviceId, size_t sizeBytes)
Copies memory from one device to memory on another device.
- `hipError_t hipMemcpyPeerAsync` (void *dst, int dstDeviceId, const void *src, int srcDevice, size_t sizeBytes, hipStream_t stream __dparm(0))
Copies memory from one device to memory on another device.

4.9.1 Detailed Description

Warning

PeerToPeer support is experimental. This section describes the PeerToPeer device memory access functions of HIP runtime API.

4.9.2 Function Documentation

4.9.2.1 hipDeviceCanAccessPeer()

```
hipError_t hipDeviceCanAccessPeer (
    int * canAccessPeer,
    int deviceId,
    int peerDeviceId )
```

Determine if a device can access a peer's memory.

Parameters

out	<i>canAccessPeer</i>	Returns the peer access capability (0 or 1)
in	<i>device</i>	- device from where memory may be accessed.
in	<i>peerDevice</i>	- device where memory is physically located

Returns "1" in *canAccessPeer* if the specified *device* is capable of directly accessing memory physically located on *peerDevice* , or "0" if not.

Returns "0" in *canAccessPeer* if *deviceId == peerDeviceId*, and both are valid devices : a device is not a peer of itself.

Returns

#hipSuccess,
 #hipErrorInvalidDevice if deviceId or peerDeviceId are not valid devices

4.9.2.2 hipDeviceDisablePeerAccess()

```
hipError_t hipDeviceDisablePeerAccess (
    int peerDeviceId )
```

Disable direct access from current device's virtual address space to memory allocations physically located on a peer device.

Returns hipErrorPeerAccessNotEnabled if direct access to memory on peerDevice has not yet been enabled from the current device.

Parameters

in	<i>peer↔ DeviceId</i>	
----	---------------------------	--

Returns

#hipSuccess, #hipErrorPeerAccessNotEnabled

4.9.2.3 hipDeviceEnablePeerAccess()

```
hipError_t hipDeviceEnablePeerAccess (
    int peerDeviceId,
    unsigned int flags )
```

Enable direct access from current device's virtual address space to memory allocations physically located on a peer device.

Memory which already allocated on peer device will be mapped into the address space of the current device. In addition, all future memory allocations on peerDeviceId will be mapped into the address space of the current device when the memory is allocated. The peer memory remains accessible from the current device until a call to hip↔DeviceDisablePeerAccess or hipDeviceReset.

Parameters

in	<i>peer↔ DeviceId</i>	
in	<i>flags</i>	Returns #hipSuccess, #hipErrorInvalidDevice, #hipErrorInvalidValue,

Returns

#hipErrorPeerAccessAlreadyEnabled if peer access is already enabled for this device.

4.9.2.4 hipMemcpyPeer()

```
hipError_t hipMemcpyPeer (
    void * dst,
    int dstDeviceId,
    const void * src,
    int srcDeviceId,
    size_t sizeBytes )
```

Copies memory from one device to memory on another device.

Parameters

out	<i>dst</i>	- Destination device pointer.
in	<i>dst</i> ↔ <i>DeviceId</i>	- Destination device
in	<i>src</i>	- Source device pointer
in	<i>src</i> ↔ <i>DeviceId</i>	- Source device
in	<i>sizeBytes</i>	- Size of memory copy in bytes

Returns

#hipSuccess, #hipErrorInvalidValue, #hipErrorInvalidDevice

4.9.2.5 hipMemcpyPeerAsync()

```
hipError_t hipMemcpyPeerAsync (
    void * dst,
    int dstDeviceId,
    const void * src,
    int srcDevice,
    size_t sizeBytes,
    hipStream_t stream __dparm0 )
```

Copies memory from one device to memory on another device.

Parameters

out	<i>dst</i>	- Destination device pointer.
in	<i>dstDevice</i>	- Destination device
in	<i>src</i>	- Source device pointer
in	<i>srcDevice</i>	- Source device
in	<i>sizeBytes</i>	- Size of memory copy in bytes
in	<i>stream</i>	- Stream identifier

Returns

#hipSuccess, #hipErrorInvalidValue, #hipErrorInvalidDevice

4.9.2.6 hipMemGetAddressRange()

```
hipError_t hipMemGetAddressRange (
    hipDeviceptr_t * pbase,
    size_t * psize,
    hipDeviceptr_t dptr )
```

Get information on memory allocations.

Parameters

out	<i>pbase</i>	- BAse pointer address
out	<i>psize</i>	- Size of allocation
in	<i>dptr</i> -	Device Pointer

Returns

`#hipSuccess`, `#hipErrorInvalidDevicePointer`

See also

[hipCtxCreate](#), [hipCtxDestroy](#), [hipCtxGetFlags](#), [hipCtxPopCurrent](#), [hipCtxGetCurrent](#), [hipCtxSetCurrent](#), [hipCtxPushCurrent](#), [hipCtxSetCacheConfig](#), [hipCtxSynchronize](#), [hipCtxGetDevice](#)

4.10 Context Management

Modules

- [Context Management \[Deprecated\]](#)

Functions

- `hipError_t hipDevicePrimaryCtxGetState` (`hipDevice_t dev`, `unsigned int *flags`, `int *active`)
Get the state of the primary context.
- `hipError_t hipDevicePrimaryCtxRelease` (`hipDevice_t dev`)
Release the primary context on the GPU.
- `hipError_t hipDevicePrimaryCtxRetain` (`hipCtx_t *pctx`, `hipDevice_t dev`)
Retain the primary context on the GPU.
- `hipError_t hipDevicePrimaryCtxReset` (`hipDevice_t dev`)
Resets the primary context on the GPU.
- `hipError_t hipDevicePrimaryCtxSetFlags` (`hipDevice_t dev`, `unsigned int flags`)
Set flags for the primary context.

4.10.1 Detailed Description

This section describes the context management functions of HIP runtime API.

4.10.2 Function Documentation

4.10.2.1 `hipDevicePrimaryCtxGetState()`

```
hipError_t hipDevicePrimaryCtxGetState (
    hipDevice_t dev,
    unsigned int * flags,
    int * active )
```

Get the state of the primary context.

Parameters

in	<i>Device</i>	to get primary context flags for
out	<i>Pointer</i>	to store flags
out	<i>Pointer</i>	to store context state; 0 = inactive, 1 = active

Returns

`#hipSuccess`

See also

[hipCtxCreate](#), [hipCtxDestroy](#), [hipCtxGetFlags](#), [hipCtxPopCurrent](#), [hipCtxGetCurrent](#), [hipCtxSetCurrent](#), [hipCtxPushCurrent](#), [hipCtxSetCacheConfig](#), [hipCtxSynchronize](#), [hipCtxGetDevice](#)

4.10.2.2 `hipDevicePrimaryCtxRelease()`

```
hipError_t hipDevicePrimaryCtxRelease (
    hipDevice_t dev )
```

Release the primary context on the GPU.

Parameters

in	<i>Device</i>	which primary context is released
----	---------------	-----------------------------------

Returns

#hipSuccess

See also

[hipCtxCreate](#), [hipCtxDestroy](#), [hipCtxGetFlags](#), [hipCtxPopCurrent](#), [hipCtxGetCurrent](#), [hipCtxSetCurrent](#), [hipCtxPushCurrent](#), [hipCtxSetCacheConfig](#), [hipCtxSynchronize](#), [hipCtxGetDevice](#)

Warning

This function return #hipSuccess though doesn't release the primaryCtx by design on HIP/HCC path.

4.10.2.3 hipDevicePrimaryCtxReset()

```
hipError_t hipDevicePrimaryCtxReset (
    hipDevice_t dev )
```

Resets the primary context on the GPU.

Parameters

in	<i>Device</i>	which primary context is reset
----	---------------	--------------------------------

Returns

#hipSuccess

See also

[hipCtxCreate](#), [hipCtxDestroy](#), [hipCtxGetFlags](#), [hipCtxPopCurrent](#), [hipCtxGetCurrent](#), [hipCtxSetCurrent](#), [hipCtxPushCurrent](#), [hipCtxSetCacheConfig](#), [hipCtxSynchronize](#), [hipCtxGetDevice](#)

4.10.2.4 hipDevicePrimaryCtxRetain()

```
hipError_t hipDevicePrimaryCtxRetain (
    hipCtx_t * pctx,
    hipDevice_t dev )
```

Retain the primary context on the GPU.

Parameters

out	<i>Returned</i>	context handle of the new context
in	<i>Device</i>	which primary context is released

Returns

#hipSuccess

See also

[hipCtxCreate](#), [hipCtxDestroy](#), [hipCtxGetFlags](#), [hipCtxPopCurrent](#), [hipCtxGetCurrent](#), [hipCtxSetCurrent](#), [hipCtxPushCurrent](#), [hipCtxSetCacheConfig](#), [hipCtxSynchronize](#), [hipCtxGetDevice](#)

4.10.2.5 hipDevicePrimaryCtxSetFlags()

```
hipError_t hipDevicePrimaryCtxSetFlags (
    hipDevice_t dev,
    unsigned int flags )
```

Set flags for the primary context.

Parameters

in	<i>Device</i>	for which the primary context flags are set
in	<i>New</i>	flags for the device

Returns

#hipSuccess, #hipErrorContextAlreadyInUse

See also

[hipCtxCreate](#), [hipCtxDestroy](#), [hipCtxGetFlags](#), [hipCtxPopCurrent](#), [hipCtxGetCurrent](#), [hipCtxSetCurrent](#), [hipCtxPushCurrent](#), [hipCtxSetCacheConfig](#), [hipCtxSynchronize](#), [hipCtxGetDevice](#)

4.11 Module Management

Functions

- hipError_t [hipModuleLoad](#) (hipModule_t *module, const char *fname)
Loads code object from file into a hipModule_t.
- hipError_t [hipModuleUnload](#) (hipModule_t module)
Frees the module.
- hipError_t [hipModuleGetFunction](#) (hipFunction_t *function, hipModule_t module, const char *kname)
Function with kname will be extracted if present in module.
- hipError_t [hipFuncGetAttributes](#) (struct [hipFuncAttributes](#) *attr, const void *func)
Find out attributes for a given function.
- hipError_t [hipFuncGetAttribute](#) (int *value, hipFunction_attribute attrib, hipFunction_t hfunc)
Find out a specific attribute for a given function.
- hipError_t [hipModuleGetTexRef](#) ([textureReference](#) **texRef, hipModule_t hmod, const char *name)
returns the handle of the texture reference with the name from the module.
- hipError_t [hipModuleLoadData](#) (hipModule_t *module, const void *image)
builds module from code object which resides in host memory. Image is pointer to that location.
- hipError_t [hipModuleLoadDataEx](#) (hipModule_t *module, const void *image, unsigned int numOptions, hipJitOption *options, void **optionValues)
builds module from code object which resides in host memory. Image is pointer to that location. Options are not used. hipModuleLoadData is called.
- hipError_t [hipModuleLaunchKernel](#) (hipFunction_t f, unsigned int gridDimX, unsigned int gridDimY, unsigned int gridDimZ, unsigned int blockDimX, unsigned int blockDimY, unsigned int blockDimZ, unsigned int sharedMemBytes, hipStream_t stream, void **kernelParams, void **extra)
launches kernel f with launch parameters and shared memory on stream with arguments passed to kernelparams or extra
- hipError_t [hipLaunchCooperativeKernel](#) (const void *f, [dim3](#) gridDim, [dim3](#) blockDimX, void **kernelParams, unsigned int sharedMemBytes, hipStream_t stream)
launches kernel f with launch parameters and shared memory on stream with arguments passed to kernelparams or extra, where thread blocks can cooperate and synchronize as they execute
- hipError_t [hipLaunchCooperativeKernelMultiDevice](#) ([hipLaunchParams](#) *launchParamsList, int numDevices, unsigned int flags)
Launches kernels on multiple devices where thread blocks can cooperate and synchronize as they execute.
- hipError_t [hipExtLaunchMultiKernelMultiDevice](#) ([hipLaunchParams](#) *launchParamsList, int numDevices, unsigned int flags)
Launches kernels on multiple devices and guarantees all specified kernels are dispatched on respective streams before enqueueing any other work on the specified streams from any other threads.
- HIP_PUBLIC_API hipError_t [hipExtModuleLaunchKernel](#) (hipFunction_t f, uint32_t globalWorkSizeX, uint32_t globalWorkSizeY, uint32_t globalWorkSizeZ, uint32_t localWorkSizeX, uint32_t localWorkSizeY, uint32_t localWorkSizeZ, size_t sharedMemBytes, hipStream_t hStream, void **kernelParams, void **extra, hipEvent_t startEvent=nullptr, hipEvent_t stopEvent=nullptr, uint32_t flags=0)
launches kernel f with launch parameters and shared memory on stream with arguments passed to kernelparams or extra
- HIP_PUBLIC_API hipError_t [hipHccModuleLaunchKernel](#) (hipFunction_t f, uint32_t globalWorkSizeX, uint32_t globalWorkSizeY, uint32_t globalWorkSizeZ, uint32_t localWorkSizeX, uint32_t localWorkSizeY, uint32_t localWorkSizeZ, size_t sharedMemBytes, hipStream_t hStream, void **kernelParams, void **extra, hipEvent_t startEvent=nullptr, hipEvent_t stopEvent=nullptr) [__attribute__\(\(deprecated\("use hipExtModuleLaunchKernel instead"\)\)\)](#)

4.11.1 Detailed Description

This section describes the module management functions of HIP runtime API.

4.11.2 Function Documentation

4.11.2.1 hipExtLaunchMultiKernelMultiDevice()

```
hipError_t hipExtLaunchMultiKernelMultiDevice (
    hipLaunchParams * launchParamsList,
    int numDevices,
    unsigned int flags )
```

Launches kernels on multiple devices and guarantees all specified kernels are dispatched on respective streams before enqueueing any other work on the specified streams from any other threads.

Parameters

in	<i>hipLaunchParams</i>	List of launch parameters, one per device.
in	<i>numDevices</i>	Size of the launchParamsList array.
in	<i>flags</i>	Flags to control launch behavior.

Returns

hipSuccess, hipInvalidDevice, hipErrorNotInitialized, hipErrorInvalidValue

4.11.2.2 hipExtModuleLaunchKernel()

```
HIP_PUBLIC_API hipError_t hipExtModuleLaunchKernel (
    hipFunction_t f,
    uint32_t globalWorkSizeX,
    uint32_t globalWorkSizeY,
    uint32_t globalWorkSizeZ,
    uint32_t localWorkSizeX,
    uint32_t localWorkSizeY,
    uint32_t localWorkSizeZ,
    size_t sharedMemBytes,
    hipStream_t hStream,
    void ** kernelParams,
    void ** extra,
    hipEvent_t startEvent = nullptr,
    hipEvent_t stopEvent = nullptr,
    uint32_t flags = 0 )
```

launches kernel f with launch parameters and shared memory on stream with arguments passed to kernelparams or extra

Parameters

	<i>[in]</i>	f Kernel to launch.
in	<i>gridDimX</i>	X grid dimension specified in work-items
in	<i>gridDimY</i>	Y grid dimension specified in work-items
in	<i>gridDimZ</i>	Z grid dimension specified in work-items
in	<i>blockDimX</i>	X block dimensions specified in work-items
in	<i>blockDimY</i>	Y grid dimension specified in work-items
in	<i>blockDimZ</i>	Z grid dimension specified in work-items
in	<i>sharedMemBytes</i>	Amount of dynamic shared memory to allocate for this kernel. The HIP-Clang compiler provides support for extern shared declarations
in	<i>stream</i>	Stream where the kernel should be dispatched. May be 0, in which case the default stream is used with associated synchronization rules.

Parameters

in	<i>kernelParams</i>	
in	<i>extra</i>	Pointer to kernel arguments. These are passed directly to the kernel and must be in the memory layout and alignment expected by the kernel.
in	<i>startEvent</i>	If non-null, specified event will be updated to track the start time of the kernel launch. The event must be created before calling this API.
in	<i>stopEvent</i>	If non-null, specified event will be updated to track the stop time of the kernel launch. The event must be created before calling this API.

Returns

hipSuccess, hipInvalidDevice, hipErrorNotInitialized, hipErrorInvalidValue

Warning

kernelParams argument is not yet implemented in HIP. Please use extra instead. Please refer to [hip_porting_driver_api.md](#) for sample usage. HIP/ROCm actually updates the start event when the associated kernel completes.

4.11.2.3 hipFuncGetAttribute()

```
hipError_t hipFuncGetAttribute (
    int * value,
    hipFunction_attribute attrib,
    hipFunction_t hfunc )
```

Find out a specific attribute for a given function.

Parameters

out	<i>value</i>	
in	<i>attrib</i>	
in	<i>hfunc</i>	

Returns

hipSuccess, hipErrorInvalidValue, hipErrorInvalidDeviceFunction

4.11.2.4 hipFuncGetAttributes()

```
hipError_t hipFuncGetAttributes (
    struct hipFuncAttributes * attr,
    const void * func )
```

Find out attributes for a given function.

Parameters

out	<i>attr</i>	
in	<i>func</i>	

Returns

hipSuccess, hipErrorInvalidValue, hipErrorInvalidDeviceFunction

4.11.2.5 hipLaunchCooperativeKernel()

```
hipError_t hipLaunchCooperativeKernel (
    const void * f,
    dim3 gridDim,
    dim3 blockDimX,
    void ** kernelParams,
    unsigned int sharedMemBytes,
    hipStream_t stream )
```

launches kernel f with launch parameters and shared memory on stream with arguments passed to kernelparams or extra, where thread blocks can cooperate and synchronize as they execute

Parameters

in	<i>f</i>	Kernel to launch.
in	<i>gridDim</i>	Grid dimensions specified as multiple of blockDim.
in	<i>blockDim</i>	Block dimensions specified in work-items
in	<i>kernelParams</i>	A list of kernel arguments
in	<i>sharedMemBytes</i>	Amount of dynamic shared memory to allocate for this kernel. The HIP-Clang compiler provides support for extern shared declarations.
in	<i>stream</i>	Stream where the kernel should be dispatched. May be 0, in which case the default stream is used with associated synchronization rules.

Returns

hipSuccess, hipInvalidDevice, hipErrorNotInitialized, hipErrorInvalidValue, hipErrorCooperativeLaunchToo↵
Large

4.11.2.6 hipLaunchCooperativeKernelMultiDevice()

```
hipError_t hipLaunchCooperativeKernelMultiDevice (
    hipLaunchParams * launchParamsList,
    int numDevices,
    unsigned int flags )
```

Launches kernels on multiple devices where thread blocks can cooperate and synchronize as they execute.

Parameters

in	<i>hipLaunchParams</i>	List of launch parameters, one per device.
in	<i>numDevices</i>	Size of the launchParamsList array.
in	<i>flags</i>	Flags to control launch behavior.

Returns

hipSuccess, hipInvalidDevice, hipErrorNotInitialized, hipErrorInvalidValue, hipErrorCooperativeLaunchToo↵
Large

4.11.2.7 hipModuleGetFunction()

```
hipError_t hipModuleGetFunction (
    hipFunction_t * function,
    hipModule_t module,
    const char * kname )
```

Function with kname will be extracted if present in module.

Parameters

in	<i>module</i>	
in	<i>kname</i>	
out	<i>function</i>	

Returns

hipSuccess, hipErrorInvalidValue, hipErrorInvalidContext, hipErrorNotInitialized, hipErrorNotFound,

4.11.2.8 hipModuleGetTexRef()

```
hipError_t hipModuleGetTexRef (
    textureReference ** texRef,
    hipModule_t hmod,
    const char * name )
```

returns the handle of the texture reference with the name from the module.

Parameters

in	<i>hmod</i>	
in	<i>name</i>	
out	<i>texRef</i>	

Returns

hipSuccess, hipErrorNotInitialized, hipErrorNotFound, hipErrorInvalidValue

4.11.2.9 hipModuleLaunchKernel()

```
hipError_t hipModuleLaunchKernel (
    hipFunction_t f,
    unsigned int gridDimX,
    unsigned int gridDimY,
    unsigned int gridDimZ,
    unsigned int blockDimX,
    unsigned int blockDimY,
    unsigned int blockDimZ,
    unsigned int sharedMemBytes,
    hipStream_t stream,
```

```
void ** kernelParams,
void ** extra )
```

launches kernel `f` with launch parameters and shared memory on stream with arguments passed to `kernelParams` or `extra`

Parameters

in	<i>f</i>	Kernel to launch.
in	<i>gridDimX</i>	X grid dimension specified as multiple of <code>blockDimX</code> .
in	<i>gridDimY</i>	Y grid dimension specified as multiple of <code>blockDimY</code> .
in	<i>gridDimZ</i>	Z grid dimension specified as multiple of <code>blockDimZ</code> .
in	<i>blockDimX</i>	X block dimensions specified in work-items
in	<i>blockDimY</i>	Y grid dimension specified in work-items
in	<i>blockDimZ</i>	Z grid dimension specified in work-items
in	<i>sharedMemBytes</i>	Amount of dynamic shared memory to allocate for this kernel. The HIP-Clang compiler provides support for extern shared declarations.
in	<i>stream</i>	Stream where the kernel should be dispatched. May be 0, in which case the default stream is used with associated synchronization rules.
in	<i>kernelParams</i>	
in	<i>extra</i>	Pointer to kernel arguments. These are passed directly to the kernel and must be in the memory layout and alignment expected by the kernel.

Returns

`hipSuccess`, `hipInvalidDevice`, `hipErrorNotInitialized`, `hipErrorInvalidValue`

Warning

`kernelParams` argument is not yet implemented in HIP. Please use `extra` instead. Please refer to `hip_porting↵_driver_api.md` for sample usage.

4.11.2.10 hipModuleLoad()

```
hipError_t hipModuleLoad (
    hipModule_t * module,
    const char * fname )
```

Loads code object from file into a `hipModule_t`.

Parameters

in	<i>fname</i>	
out	<i>module</i>	

Returns

`hipSuccess`, `hipErrorInvalidValue`, `hipErrorInvalidContext`, `hipErrorFileNotFound`, `hipErrorOutOfMemory`, `hip↵ErrorSharedObjectInitFailed`, `hipErrorNotInitialized`

4.11.2.11 hipModuleLoadData()

```
hipError_t hipModuleLoadData (
    hipModule_t * module,
    const void * image )
```

builds module from code object which resides in host memory. Image is pointer to that location.

Parameters

in	<i>image</i>	
out	<i>module</i>	

Returns

hipSuccess, hipErrorNotInitialized, hipErrorOutOfMemory, hipErrorNotInitialized

4.11.2.12 hipModuleLoadDataEx()

```
hipError_t hipModuleLoadDataEx (
    hipModule_t * module,
    const void * image,
    unsigned int numOptions,
    hipJitOption * options,
    void ** optionValues )
```

builds module from code object which resides in host memory. Image is pointer to that location. Options are not used. hipModuleLoadData is called.

Parameters

in	<i>image</i>	
out	<i>module</i>	
in	<i>number</i>	of options
in	<i>options</i>	for JIT
in	<i>option</i>	values for JIT

Returns

hipSuccess, hipErrorNotInitialized, hipErrorOutOfMemory, hipErrorNotInitialized

4.11.2.13 hipModuleUnload()

```
hipError_t hipModuleUnload (
    hipModule_t module )
```

Frees the module.

Parameters

in	<i>module</i>	
----	---------------	--

Returns

hipSuccess, hipInvalidValue module is freed and the code objects associated with it are destroyed

4.12 Occupancy

Functions

- hipError_t [hipModuleOccupancyMaxPotentialBlockSize](#) (int *gridSize, int *blockSize, hipFunction_t f, size_t dynSharedMemPerBlk, int blockSizeLimit)
determine the grid and block sizes to achieves maximum occupancy for a kernel
- hipError_t [hipModuleOccupancyMaxPotentialBlockSizeWithFlags](#) (int *gridSize, int *blockSize, hipFunction_t f, size_t dynSharedMemPerBlk, int blockSizeLimit, unsigned int flags)
determine the grid and block sizes to achieves maximum occupancy for a kernel
- hipError_t [hipModuleOccupancyMaxActiveBlocksPerMultiprocessor](#) (int *numBlocks, hipFunction_t f, int blockSize, size_t dynSharedMemPerBlk)
Returns occupancy for a device function.
- hipError_t [hipModuleOccupancyMaxActiveBlocksPerMultiprocessorWithFlags](#) (int *numBlocks, hipFunction_t f, int blockSize, size_t dynSharedMemPerBlk, unsigned int flags)
Returns occupancy for a device function.
- hipError_t [hipOccupancyMaxActiveBlocksPerMultiprocessor](#) (int *numBlocks, const void *f, int blockSize, size_t dynSharedMemPerBlk)
Returns occupancy for a device function.
- hipError_t [hipOccupancyMaxActiveBlocksPerMultiprocessorWithFlags](#) (int *numBlocks, const void *f, int blockSize, size_t dynSharedMemPerBlk, unsigned int flags __dparm(hipOccupancyDefault))
Returns occupancy for a device function.
- hipError_t [hipOccupancyMaxPotentialBlockSize](#) (int *gridSize, int *blockSize, const void *f, size_t dynSharedMemPerBlk, int blockSizeLimit)
determine the grid and block sizes to achieves maximum occupancy for a kernel

4.12.1 Detailed Description

This section describes the occupancy functions of HIP runtime API.

4.12.2 Function Documentation

4.12.2.1 hipModuleOccupancyMaxActiveBlocksPerMultiprocessor()

```
hipError_t hipModuleOccupancyMaxActiveBlocksPerMultiprocessor (
    int * numBlocks,
    hipFunction_t f,
    int blockSize,
    size_t dynSharedMemPerBlk )
```

Returns occupancy for a device function.

Parameters

out	<i>numBlocks</i>	Returned occupancy
in	<i>func</i>	Kernel function (hipFunction) for which occupancy is calculated
in	<i>blockSize</i>	Block size the kernel is intended to be launched with
in	<i>dynSharedMemPerBlk</i>	dynamic shared memory usage (in bytes) intended for each block

4.12.2.2 hipModuleOccupancyMaxActiveBlocksPerMultiprocessorWithFlags()

```
hipError_t hipModuleOccupancyMaxActiveBlocksPerMultiprocessorWithFlags (
    int * numBlocks,
```

```

hipFunction_t f,
int blockSize,
size_t dynSharedMemPerBlk,
unsigned int flags )

```

Returns occupancy for a device function.

Parameters

out	<i>numBlocks</i>	Returned occupancy
in	<i>f</i>	Kernel function(hipFunction_t) for which occupancy is calculated
in	<i>blockSize</i>	Block size the kernel is intended to be launched with
in	<i>dynSharedMemPerBlk</i>	dynamic shared memory usage (in bytes) intended for each block
in	<i>flags</i>	Extra flags for occupancy calculation (only default supported)

4.12.2.3 hipModuleOccupancyMaxPotentialBlockSize()

```

hipError_t hipModuleOccupancyMaxPotentialBlockSize (
    int * gridSize,
    int * blockSize,
    hipFunction_t f,
    size_t dynSharedMemPerBlk,
    int blockSizeLimit )

```

determine the grid and block sizes to achieves maximum occupancy for a kernel

Parameters

out	<i>gridSize</i>	minimum grid size for maximum potential occupancy
out	<i>blockSize</i>	block size for maximum potential occupancy
in	<i>f</i>	kernel function for which occupancy is calculated
in	<i>dynSharedMemPerBlk</i>	dynamic shared memory usage (in bytes) intended for each block
in	<i>blockSizeLimit</i>	the maximum block size for the kernel, use 0 for no limit

Returns

hipSuccess, hipInvalidDevice, hipErrorInvalidValue

4.12.2.4 hipModuleOccupancyMaxPotentialBlockSizeWithFlags()

```

hipError_t hipModuleOccupancyMaxPotentialBlockSizeWithFlags (
    int * gridSize,
    int * blockSize,
    hipFunction_t f,
    size_t dynSharedMemPerBlk,
    int blockSizeLimit,
    unsigned int flags )

```

determine the grid and block sizes to achieves maximum occupancy for a kernel

Parameters

out	<i>gridSize</i>	minimum grid size for maximum potential occupancy
out	<i>blockSize</i>	block size for maximum potential occupancy
in	<i>f</i>	kernel function for which occupancy is calculated

Parameters

in	<i>dynSharedMemPerBlk</i>	dynamic shared memory usage (in bytes) intended for each block
in	<i>blockSizeLimit</i>	the maximum block size for the kernel, use 0 for no limit
in	<i>flags</i>	Extra flags for occupancy calculation (only default supported)

Returns

hipSuccess, hipInvalidDevice, hipErrorInvalidValue

4.12.2.5 hipOccupancyMaxActiveBlocksPerMultiprocessor()

```
hipError_t hipOccupancyMaxActiveBlocksPerMultiprocessor (
    int * numBlocks,
    const void * f,
    int blockSize,
    size_t dynSharedMemPerBlk )
```

Returns occupancy for a device function.

Parameters

out	<i>numBlocks</i>	Returned occupancy
in	<i>func</i>	Kernel function for which occupancy is calculated
in	<i>blockSize</i>	Block size the kernel is intended to be launched with
in	<i>dynSharedMemPerBlk</i>	dynamic shared memory usage (in bytes) intended for each block

4.12.2.6 hipOccupancyMaxActiveBlocksPerMultiprocessorWithFlags()

```
hipError_t hipOccupancyMaxActiveBlocksPerMultiprocessorWithFlags (
    int * numBlocks,
    const void * f,
    int blockSize,
    size_t dynSharedMemPerBlk,
    unsigned int flags __dparmhipOccupancyDefault )
```

Returns occupancy for a device function.

Parameters

out	<i>numBlocks</i>	Returned occupancy
in	<i>f</i>	Kernel function for which occupancy is calculated
in	<i>blockSize</i>	Block size the kernel is intended to be launched with
in	<i>dynSharedMemPerBlk</i>	dynamic shared memory usage (in bytes) intended for each block
in	<i>flags</i>	Extra flags for occupancy calculation (currently ignored)

4.12.2.7 hipOccupancyMaxPotentialBlockSize()

```
hipError_t hipOccupancyMaxPotentialBlockSize (
    int * gridSize,
    int * blockSize,
    const void * f,
```

```
size_t dynSharedMemPerBlk,  
int blockSizeLimit )
```

determine the grid and block sizes to achieves maximum occupancy for a kernel

Parameters

out	<i>gridSize</i>	minimum grid size for maximum potential occupancy
out	<i>blockSize</i>	block size for maximum potential occupancy
in	<i>f</i>	kernel function for which occupancy is calulated
in	<i>dynSharedMemPerBlk</i>	dynamic shared memory usage (in bytes) intended for each block
in	<i>blockSizeLimit</i>	the maximum block size for the kernel, use 0 for no limit

Returns

hipSuccess, hipInvalidDevice, hipErrorInvalidValue

4.13 Profiler Control[Deprecated]

Functions

- `hipError_t hipProfilerStart ()`
Start recording of profiling information When using this API, start the profiler with profiling disabled. (--startdisabled)
- `hipError_t hipProfilerStop ()`
Stop recording of profiling information. When using this API, start the profiler with profiling disabled. (--startdisabled)

4.13.1 Detailed Description

This section describes the profiler control functions of HIP runtime API.

Warning

The `cudaProfilerInitialize` API format for "configFile" is not supported.

4.13.2 Function Documentation

4.13.2.1 hipProfilerStart()

```
hipError_t hipProfilerStart ( )
```

Start recording of profiling information When using this API, start the profiler with profiling disabled. (--startdisabled)

Warning

: `hipProfilerStart` API is under development.

4.13.2.2 hipProfilerStop()

```
hipError_t hipProfilerStop ( )
```

Stop recording of profiling information. When using this API, start the profiler with profiling disabled. (--startdisabled)

Warning

: `hipProfilerStop` API is under development.

4.14 Launch API to support the triple-chevron syntax

Functions

- hipError_t [hipConfigureCall](#) (dim3 gridDim, dim3 blockDim, size_t sharedMem __dparm(0), hipStream_t stream __dparm(0))
Configure a kernel launch.
- hipError_t [hipSetupArgument](#) (const void *arg, size_t size, size_t offset)
Set a kernel argument.
- hipError_t [hipLaunchByPtr](#) (const void *func)
Launch a kernel.
- hipError_t [__hipPushCallConfiguration](#) (dim3 gridDim, dim3 blockDim, size_t sharedMem __dparm(0), hipStream_t stream __dparm(0))
Push configuration of a kernel launch.
- hipError_t [__hipPopCallConfiguration](#) (dim3 *gridDim, dim3 *blockDim, size_t *sharedMem, hipStream_t *stream)
Pop configuration of a kernel launch.
- hipError_t [hipLaunchKernel](#) (const void *function_address, dim3 numBlocks, dim3 dimBlocks, void **args, size_t sharedMemBytes __dparm(0), hipStream_t stream __dparm(0))
C compliant kernel launch API.
- hipError_t [hipDrvMemcpy2DUnaligned](#) (const hip_Memcpy2D *pCopy)
- hipError_t [hipExtLaunchKernel](#) (const void *function_address, dim3 numBlocks, dim3 dimBlocks, void **args, size_t sharedMemBytes, hipStream_t stream, hipEvent_t startEvent, hipEvent_t stopEvent, int flags)

4.14.1 Detailed Description

This section describes the API to support the triple-chevron syntax.

4.14.2 Function Documentation

4.14.2.1 __hipPopCallConfiguration()

```
hipError_t __hipPopCallConfiguration (
    dim3 * gridDim,
    dim3 * blockDim,
    size_t * sharedMem,
    hipStream_t * stream )
```

Pop configuration of a kernel launch.

Parameters

out	<i>gridDim</i>	grid dimension specified as multiple of blockDim.
out	<i>blockDim</i>	block dimensions specified in work-items
out	<i>sharedMem</i>	Amount of dynamic shared memory to allocate for this kernel. The HIP-Clang compiler provides support for extern shared declarations.
out	<i>stream</i>	Stream where the kernel should be dispatched. May be 0, in which case the default stream is used with associated synchronization rules.

Returns

hipSuccess, hipInvalidDevice, hipErrorNotInitialized, hipErrorInvalidValue

4.14.2.2 __hipPushCallConfiguration()

```
hipError_t __hipPushCallConfiguration (
    dim3 gridDim,
    dim3 blockDim,
    size_t sharedMem __dparm0,
    hipStream_t stream __dparm0 )
```

Push configuration of a kernel launch.

Parameters

in	<i>gridDim</i>	grid dimension specified as multiple of blockDim.
in	<i>blockDim</i>	block dimensions specified in work-items
in	<i>sharedMem</i>	Amount of dynamic shared memory to allocate for this kernel. The HIP-Clang compiler provides support for extern shared declarations.
in	<i>stream</i>	Stream where the kernel should be dispatched. May be 0, in which case the default stream is used with associated synchronization rules.

Returns

hipSuccess, hipInvalidDevice, hipErrorNotInitialized, hipErrorInvalidValue

4.14.2.3 hipConfigureCall()

```
hipError_t hipConfigureCall (
    dim3 gridDim,
    dim3 blockDim,
    size_t sharedMem __dparm0,
    hipStream_t stream __dparm0 )
```

Configure a kernel launch.

Parameters

in	<i>gridDim</i>	grid dimension specified as multiple of blockDim.
in	<i>blockDim</i>	block dimensions specified in work-items
in	<i>sharedMem</i>	Amount of dynamic shared memory to allocate for this kernel. The HIP-Clang compiler provides support for extern shared declarations.
in	<i>stream</i>	Stream where the kernel should be dispatched. May be 0, in which case the default stream is used with associated synchronization rules.

Returns

hipSuccess, hipInvalidDevice, hipErrorNotInitialized, hipErrorInvalidValue

4.14.2.4 hipDrvMemcpy2DUnaligned()

```
hipError_t hipDrvMemcpy2DUnaligned (
    const hip_Memcpy2D * pCopy )
```

Copies memory for 2D arrays.

Parameters

<i>pCopy</i>	- Parameters for the memory copy
--------------	----------------------------------

Returns

#hipSuccess, #hipErrorInvalidValue

4.14.2.5 hipLaunchByPtr()

```
hipError_t hipLaunchByPtr (
    const void * func )
```

Launch a kernel.

Parameters

in	<i>func</i>	Kernel to launch.
----	-------------	-------------------

Returns

hipSuccess, hipInvalidDevice, hipErrorNotInitialized, hipErrorInvalidValue

4.14.2.6 hipLaunchKernel()

```
hipError_t hipLaunchKernel (
    const void * function_address,
    dim3 numBlocks,
    dim3 dimBlocks,
    void ** args,
    size_t sharedMemBytes __dparm0,
    hipStream_t stream __dparm0 )
```

C compliant kernel launch API.

Parameters

in	<i>function_address</i>	- kernel stub function pointer.
in	<i>numBlocks</i>	- number of blocks
in	<i>dimBlocks</i>	- dimension of a block
in	<i>args</i>	- kernel arguments
in	<i>sharedMemBytes</i>	- Amount of dynamic shared memory to allocate for this kernel. The HIP-Clang compiler provides support for extern shared declarations.
in	<i>stream</i>	- Stream where the kernel should be dispatched. May be 0, in which case the default stream is used with associated synchronization rules.

Returns

#hipSuccess, #hipErrorInvalidValue, hipInvalidDevice

4.14.2.7 hipSetupArgument()

```
hipError_t hipSetupArgument (
    const void * arg,
```

```
size_t size,  
size_t offset )
```

Set a kernel argument.

Returns

hipSuccess, hipInvalidDevice, hipErrorNotInitialized, hipErrorInvalidValue

Parameters

in	<i>arg</i>	Pointer the argument in host memory.
in	<i>size</i>	Size of the argument.
in	<i>offset</i>	Offset of the argument on the argument stack.

4.15 Texture Management

Modules

- [Texture Management \[Deprecated\]](#)

Functions

- `hipError_t hipBindTextureToMipmappedArray` (const [textureReference](#) *tex, [hipMipmappedArray_const_t](#) mipmappedArray, const [hipChannelFormatDesc](#) *desc)
- `hipError_t hipGetTextureReference` (const [textureReference](#) **texref, const void *symbol)
- `hipError_t hipCreateTextureObject` (hipTextureObject_t *pTexObject, const [hipResourceDesc](#) *pResDesc, const [hipTextureDesc](#) *pTexDesc, const struct [hipResourceViewDesc](#) *pResViewDesc)
- `hipError_t hipDestroyTextureObject` (hipTextureObject_t textureObject)
- `hipError_t hipGetChannelDesc` ([hipChannelFormatDesc](#) *desc, [hipArray_const_t](#) array)
- `hipError_t hipGetTextureObjectResourceDesc` ([hipResourceDesc](#) *pResDesc, hipTextureObject_t textureObject ↵ t textureObject)
- `hipError_t hipGetTextureObjectResourceViewDesc` (struct [hipResourceViewDesc](#) *pResViewDesc, hipTextureObject_t textureObject ↵ textureObject)
- `hipError_t hipGetTextureObjectTextureDesc` ([hipTextureDesc](#) *pTexDesc, hipTextureObject_t textureObject ↵ Object)
- `hipError_t hipTexRefGetAddress` (hipDeviceptr_t *dev_ptr, const [textureReference](#) *texRef)
- `hipError_t hipTexRefGetAddressMode` (enum hipTextureAddressMode *pam, const [textureReference](#) *texRef, int dim)
- `hipError_t hipTexRefGetFilterMode` (enum hipTextureFilterMode *pfm, const [textureReference](#) *texRef)
- `hipError_t hipTexRefGetFlags` (unsigned int *pFlags, const [textureReference](#) *texRef)
- `hipError_t hipTexRefGetFormat` (hipArray_Format *pFormat, int *pNumChannels, const [textureReference](#) *texRef)
- `hipError_t hipTexRefGetMaxAnisotropy` (int *pmaxAnsio, const [textureReference](#) *texRef)
- `hipError_t hipTexRefGetMipmapFilterMode` (enum hipTextureFilterMode *pfm, const [textureReference](#) *texRef)
- `hipError_t hipTexRefGetMipmapLevelBias` (float *pbias, const [textureReference](#) *texRef)
- `hipError_t hipTexRefGetMipmapLevelClamp` (float *pminMipmapLevelClamp, float *pmaxMipmapLevelClamp, const [textureReference](#) *texRef)
- `hipError_t hipTexRefGetMipMappedArray` ([hipMipmappedArray_t](#) *pArray, const [textureReference](#) *texRef ↵ Ref)
- `hipError_t hipTexRefSetAddress` (size_t *ByteOffset, [textureReference](#) *texRef, hipDeviceptr_t dptr, size_t ↵ bytes)
- `hipError_t hipTexRefSetAddress2D` ([textureReference](#) *texRef, const [HIP_ARRAY_DESCRIPTOR](#) *desc, hipDeviceptr_t dptr, size_t Pitch)
- `hipError_t hipTexRefSetAddressMode` ([textureReference](#) *texRef, int dim, enum hipTextureAddressMode am)
- `hipError_t hipTexRefSetArray` ([textureReference](#) *tex, [hipArray_const_t](#) array, unsigned int flags)
- `hipError_t hipTexRefSetFilterMode` ([textureReference](#) *texRef, enum hipTextureFilterMode fm)
- `hipError_t hipTexRefSetFlags` ([textureReference](#) *texRef, unsigned int Flags)
- `hipError_t hipTexRefSetFormat` ([textureReference](#) *texRef, hipArray_Format fmt, int NumPacked ↵ Components)
- `hipError_t hipTexRefSetMaxAnisotropy` ([textureReference](#) *texRef, unsigned int maxAniso)
- `hipError_t hipTexObjectCreate` (hipTextureObject_t *pTexObject, const [HIP_RESOURCE_DESC](#) *pResDesc, const [HIP_TEXTURE_DESC](#) *pTexDesc, const [HIP_RESOURCE_VIEW_DESC](#) *pResViewDesc)
- `hipError_t hipTexObjectDestroy` (hipTextureObject_t texObject)
- `hipError_t hipTexObjectGetResourceDesc` ([HIP_RESOURCE_DESC](#) *pResDesc, hipTextureObject_t texObject ↵ t texObject)
- `hipError_t hipTexObjectGetResourceViewDesc` ([HIP_RESOURCE_VIEW_DESC](#) *pResViewDesc, hipTextureObject_t texObject ↵ texObject)
- `hipError_t hipTexObjectGetTextureDesc` ([HIP_TEXTURE_DESC](#) *pTexDesc, hipTextureObject_t texObject ↵ Object)

4.15.1 Detailed Description

This section describes the texture management functions of HIP runtime API.

4.16 Global enum and defines

Classes

- struct [dim3](#)
- struct [hipLaunchParams_t](#)
- struct [hipExternalMemoryHandleDesc_st](#)
- struct [hipExternalMemoryBufferDesc_st](#)

Macros

- #define [hipStreamDefault](#) 0x00
Flags that can be used with [hipStreamCreateWithFlags](#).
- #define [hipStreamNonBlocking](#) 0x01
Stream does not implicitly synchronize with null stream.
- #define [hipEventDefault](#) 0x0
Flags that can be used with [hipEventCreateWithFlags](#):
- #define [hipEventBlockingSync](#) 0x1
Waiting will yield CPU. Power-friendly and usage-friendly but may increase latency.
- #define [hipEventDisableTiming](#) 0x2
Disable event's capability to record timing information. May improve performance.
- #define [hipEventInterprocess](#) 0x4
Event can support IPC.
- #define [hipEventReleaseToDevice](#) 0x40000000
- #define [hipEventReleaseToSystem](#) 0x80000000
- #define [hipHostMallocDefault](#) 0x0
Flags that can be used with [hipHostMalloc](#).
- #define [hipHostMallocPortable](#) 0x1
Memory is considered allocated by all contexts.
- #define [hipHostMallocMapped](#) 0x2
can be obtained with [hipHostGetDevicePointer](#).
- #define [hipHostMallocWriteCombined](#) 0x4
- #define [hipHostMallocNumaUser](#) 0x20000000
Host memory allocation will follow numa policy set by user.
- #define [hipHostMallocCoherent](#) 0x40000000
allocation.
- #define [hipHostMallocNonCoherent](#) 0x80000000
allocation.
- #define [hipMemAttachGlobal](#) 0x01
Memory can be accessed by any stream on any device.
- #define [hipMemAttachHost](#) 0x02
Memory cannot be accessed by any stream on any device.
- #define [hipMemAttachSingle](#) 0x04
the associated device
- #define [hipDeviceMallocDefault](#) 0x0
- #define [hipDeviceMallocFinegrained](#) 0x1
Memory is allocated in fine grained region of device.
- #define [hipMallocSignalMemory](#) 0x2
Memory represents a HSA signal.
- #define [hipHostRegisterDefault](#) 0x0
Flags that can be used with [hipHostRegister](#).
- #define [hipHostRegisterPortable](#) 0x1

- Memory is considered registered by all contexts.*

 - #define `hipHostRegisterMapped` 0x2
 - can be obtained with `hipHostGetDevicePointer`.*
 - #define `hipHostRegisterIoMemory` 0x4
 - Not supported.*
 - #define `hipExtHostRegisterCoarseGrained` 0x8
 - Coarse Grained host memory lock.*
 - #define `hipDeviceScheduleAuto` 0x0
 - Automatically select between Spin and Yield.*
 - #define `hipDeviceScheduleSpin` 0x1
 - may consume more power.*
 - #define `hipDeviceScheduleYield` 0x2
 - power and is friendlier to other threads in the system.*
 - #define `hipDeviceScheduleBlockingSync` 0x4
 - #define `hipDeviceScheduleMask` 0x7
 - #define `hipDeviceMapHost` 0x8
 - #define `hipDeviceLmemResizeToMax` 0x16
 - #define `hipArrayDefault` 0x00
 - Default HIP array allocation flag.*
 - #define `hipArrayLayered` 0x01
 - #define `hipArraySurfaceLoadStore` 0x02
 - #define `hipArrayCubemap` 0x04
 - #define `hipArrayTextureGather` 0x08
 - #define `hipOccupancyDefault` 0x00
 - #define `hipCooperativeLaunchMultiDeviceNoPreSync` 0x01
 - #define `hipCooperativeLaunchMultiDeviceNoPostSync` 0x02
 - #define `hipCpuDeviceld` ((int)-1)
 - #define `hipInvalidDeviceld` ((int)-2)
 - #define `hipExtAnyOrderLaunch` 0x01
 - AnyOrderLaunch of kernels.*
 - #define `hipStreamWaitValueGte` 0x0
 - #define `hipStreamWaitValueEq` 0x1
 - #define `hipStreamWaitValueAnd` 0x2
 - #define `hipStreamWaitValueNor` 0x3
 - #define `hipStreamDefault` 0x00
 - Flags that can be used with `hipStreamCreateWithFlags`.*
 - #define `hipStreamNonBlocking` 0x01
 - Stream does not implicitly synchronize with null stream.*
 - #define `hipEventDefault` 0x0
 - Flags that can be used with `hipEventCreateWithFlags`:*
 - #define `hipEventBlockingSync` 0x1
 - Waiting will yield CPU. Power-friendly and usage-friendly but may increase latency.*
 - #define `hipEventDisableTiming` 0x2
 - Disable event's capability to record timing information. May improve performance.*
 - #define `hipEventInterprocess` 0x4
 - Event can support IPC.*
 - #define `hipEventReleaseToDevice` 0x40000000
 - #define `hipEventReleaseToSystem` 0x80000000
 - #define `hipHostMallocDefault` 0x0
 - Flags that can be used with `hipHostMalloc`.*
 - #define `hipHostMallocPortable` 0x1
 - Memory is considered allocated by all contexts.*

- #define `hipHostMallocMapped` 0x2
can be obtained with `hipHostGetDevicePointer`.
- #define `hipHostMallocWriteCombined` 0x4
- #define `hipHostMallocNumaUser` 0x20000000
Host memory allocation will follow numa policy set by user.
- #define `hipHostMallocCoherent` 0x40000000
allocation.
- #define `hipHostMallocNonCoherent` 0x80000000
allocation.
- #define `hipMemAttachGlobal` 0x01
Memory can be accessed by any stream on any device.
- #define `hipMemAttachHost` 0x02
Memory cannot be accessed by any stream on any device.
- #define `hipMemAttachSingle` 0x04
the associated device
- #define `hipDeviceMallocDefault` 0x0
- #define `hipDeviceMallocFinegrained` 0x1
Memory is allocated in fine grained region of device.
- #define `hipMallocSignalMemory` 0x2
Memory represents a HSA signal.
- #define `hipHostRegisterDefault` 0x0
Flags that can be used with `hipHostRegister`.
- #define `hipHostRegisterPortable` 0x1
Memory is considered registered by all contexts.
- #define `hipHostRegisterMapped` 0x2
can be obtained with `hipHostGetDevicePointer`.
- #define `hipHostRegisterIoMemory` 0x4
Not supported.
- #define `hipExtHostRegisterCoarseGrained` 0x8
Coarse Grained host memory lock.
- #define `hipDeviceScheduleAuto` 0x0
Automatically select between Spin and Yield.
- #define `hipDeviceScheduleSpin` 0x1
may consume more power.
- #define `hipDeviceScheduleYield` 0x2
power and is friendlier to other threads in the system.
- #define `hipDeviceScheduleBlockingSync` 0x4
- #define `hipDeviceScheduleMask` 0x7
- #define `hipDeviceMapHost` 0x8
- #define `hipDeviceLmemResizeToMax` 0x16
- #define `hipArrayDefault` 0x00
Default HIP array allocation flag.
- #define `hipArrayLayered` 0x01
- #define `hipArraySurfaceLoadStore` 0x02
- #define `hipArrayCubemap` 0x04
- #define `hipArrayTextureGather` 0x08
- #define `hipOccupancyDefault` 0x00
- #define `hipCooperativeLaunchMultiDeviceNoPreSync` 0x01
- #define `hipCooperativeLaunchMultiDeviceNoPostSync` 0x02
- #define `hipCpuDeviceId` ((int)-1)
- #define `hipInvalidDeviceId` ((int)-2)
- #define `hipExtAnyOrderLaunch` 0x01

AnyOrderLaunch of kernels.

- #define **hipStreamWaitValueGte** 0x0
- #define **hipStreamWaitValueEq** 0x1
- #define **hipStreamWaitValueAnd** 0x2
- #define **hipStreamWaitValueNor** 0x3
- #define **__HIP_NODISCARD**

Typedefs

- typedef enum [hipMemoryAdvise](#) **hipMemoryAdvise**
- typedef enum [hipMemRangeAttribute](#) **hipMemRangeAttribute**
- typedef enum [hipJitOption](#) **hipJitOption**
- typedef enum [hipFuncAttribute](#) **hipFuncAttribute**
- typedef enum [hipFuncCache_t](#) **hipFuncCache_t**
- typedef enum [hipSharedMemConfig](#) **hipSharedMemConfig**
- typedef struct [dim3](#) **dim3**
- typedef struct [hipLaunchParams_t](#) **hipLaunchParams**
- typedef enum [hipExternalMemoryHandleType_enum](#) **hipExternalMemoryHandleType**
- typedef struct [hipExternalMemoryHandleDesc_st](#) **hipExternalMemoryHandleDesc**
- typedef struct [hipExternalMemoryBufferDesc_st](#) **hipExternalMemoryBufferDesc**
- typedef void * **hipExternalMemory_t**
- typedef enum [hipMemoryAdvise](#) **hipMemoryAdvise**
- typedef enum [hipMemRangeAttribute](#) **hipMemRangeAttribute**
- typedef enum [hipJitOption](#) **hipJitOption**
- typedef enum [hipFuncAttribute](#) **hipFuncAttribute**
- typedef enum [hipFuncCache_t](#) **hipFuncCache_t**
- typedef enum [hipSharedMemConfig](#) **hipSharedMemConfig**
- typedef struct [dim3](#) **dim3**
- typedef struct [hipLaunchParams_t](#) **hipLaunchParams**
- typedef enum [hipExternalMemoryHandleType_enum](#) **hipExternalMemoryHandleType**
- typedef struct [hipExternalMemoryHandleDesc_st](#) **hipExternalMemoryHandleDesc**
- typedef struct [hipExternalMemoryBufferDesc_st](#) **hipExternalMemoryBufferDesc**
- typedef void * **hipExternalMemory_t**
- typedef enum [__HIP_NODISCARD](#) [hipError_t](#) **hipError_t**
- typedef enum [hipDeviceAttribute_t](#) **hipDeviceAttribute_t**

Enumerations

- enum [hipMemoryAdvise](#) {
[hipMemAdviseSetReadMostly](#) = 1, [hipMemAdviseUnsetReadMostly](#) = 2, [hipMemAdviseSetPreferredLocation](#)
= 3, [hipMemAdviseUnsetPreferredLocation](#) = 4,
[hipMemAdviseSetAccessedBy](#) = 5, [hipMemAdviseUnsetAccessedBy](#) = 6, [hipMemAdviseSetReadMostly](#) =
1, [hipMemAdviseUnsetReadMostly](#) = 2,
[hipMemAdviseSetPreferredLocation](#) = 3, [hipMemAdviseUnsetPreferredLocation](#) = 4, [hipMemAdviseSetAccessedBy](#)
= 5, [hipMemAdviseUnsetAccessedBy](#) = 6,
hipMemAdviseSetReadMostly, **hipMemAdviseUnsetReadMostly**, **hipMemAdviseSetPreferred**↵
Location, **hipMemAdviseUnsetPreferredLocation**,
hipMemAdviseSetAccessedBy, **hipMemAdviseUnsetAccessedBy** }
- enum [hipMemRangeAttribute](#) {
[hipMemRangeAttributeReadMostly](#) = 1, [hipMemRangeAttributePreferredLocation](#) = 2, [hipMemRangeAttributeAccessedBy](#)
= 3, [hipMemRangeAttributeLastPrefetchLocation](#) = 4,
[hipMemRangeAttributeReadMostly](#) = 1, [hipMemRangeAttributePreferredLocation](#) = 2, [hipMemRangeAttributeAccessedBy](#)
= 3, [hipMemRangeAttributeLastPrefetchLocation](#) = 4 }

- enum `hipJitOption` {
`hipJitOptionMaxRegisters = 0, hipJitOptionThreadsPerBlock, hipJitOptionWallTime, hipJitOptionInfoLogBuffer,`
`hipJitOptionInfoLogBufferSizeBytes, hipJitOptionErrorLogBuffer, hipJitOptionErrorLogBufferSizeBytes, hipJitOptionOptimizationLevel,`
`hipJitOptionTargetFromContext, hipJitOptionTarget, hipJitOptionFallbackStrategy, hipJitOptionGenerateDebugInfo,`
`hipJitOptionLogVerbose, hipJitOptionGenerateLineInfo, hipJitOptionCacheMode, hipJitOptionSm3xOpt,`
`hipJitOptionFastCompile, hipJitOptionNumOptions, hipJitOptionMaxRegisters = 0, hipJitOptionThreadsPerBlock,`
`hipJitOptionWallTime, hipJitOptionInfoLogBuffer, hipJitOptionInfoLogBufferSizeBytes, hipJitOptionErrorLogBuffer,`
`hipJitOptionErrorLogBufferSizeBytes, hipJitOptionOptimizationLevel, hipJitOptionTargetFromContext, hipJitOptionTarget,`
`hipJitOptionFallbackStrategy, hipJitOptionGenerateDebugInfo, hipJitOptionLogVerbose, hipJitOptionGenerateLineInfo,`
`hipJitOptionCacheMode, hipJitOptionSm3xOpt, hipJitOptionFastCompile, hipJitOptionNumOptions }`
- enum `hipFuncAttribute` {
`hipFuncAttributeMaxDynamicSharedMemorySize = 8, hipFuncAttributePreferredSharedMemoryCarveout = 9, hipFuncAttributeMax,`
`hipFuncAttributeMaxDynamicSharedMemorySize = 8, hipFuncAttributePreferredSharedMemoryCarveout = 9, hipFuncAttributeMax }`
- enum `hipFuncCache_t` {
`hipFuncCachePreferNone, hipFuncCachePreferShared, hipFuncCachePreferL1, hipFuncCachePreferEqual,`
`hipFuncCachePreferNone, hipFuncCachePreferShared, hipFuncCachePreferL1, hipFuncCachePreferEqual }`
- enum `hipSharedMemConfig` {
`hipSharedMemBankSizeDefault, hipSharedMemBankSizeFourByte, hipSharedMemBankSizeEightByte,`
`hipSharedMemBankSizeDefault, hipSharedMemBankSizeFourByte, hipSharedMemBankSizeEightByte }`
- enum `hipExternalMemoryHandleType_enum` {
`hipExternalMemoryHandleTypeOpaqueFd = 1, hipExternalMemoryHandleTypeOpaqueWin32 = 2,`
`hipExternalMemoryHandleTypeOpaqueWin32Kmt = 3, hipExternalMemoryHandleTypeD3D12Heap = 4,`
`hipExternalMemoryHandleTypeD3D12Resource = 5, hipExternalMemoryHandleTypeD3D11Resource = 6,`
`hipExternalMemoryHandleTypeD3D11ResourceKmt = 7, hipExternalMemoryHandleTypeOpaqueFd = 1,`
`hipExternalMemoryHandleTypeOpaqueWin32 = 2, hipExternalMemoryHandleTypeOpaqueWin32Kmt = 3,`
`hipExternalMemoryHandleTypeD3D12Heap = 4, hipExternalMemoryHandleTypeD3D12Resource = 5,`
`hipExternalMemoryHandleTypeD3D11Resource = 6, hipExternalMemoryHandleTypeD3D11ResourceKmt = 7 }`
- enum `hipMemoryAdvise` {
`hipMemAdviseSetReadMostly = 1, hipMemAdviseUnsetReadMostly = 2, hipMemAdviseSetPreferredLocation = 3,`
`hipMemAdviseUnsetPreferredLocation = 4,`
`hipMemAdviseSetAccessedBy = 5, hipMemAdviseUnsetAccessedBy = 6, hipMemAdviseSetReadMostly = 1,`
`hipMemAdviseUnsetReadMostly = 2,`
`hipMemAdviseSetPreferredLocation = 3, hipMemAdviseUnsetPreferredLocation = 4, hipMemAdviseSetAccessedBy = 5,`
`hipMemAdviseUnsetAccessedBy = 6,`
`hipMemAdviseSetReadMostly, hipMemAdviseUnsetReadMostly, hipMemAdviseSetPreferredLocation, hipMemAdviseUnsetPreferredLocation,`
`hipMemAdviseSetAccessedBy, hipMemAdviseUnsetAccessedBy }`
- enum `hipMemRangeAttribute` {
`hipMemRangeAttributeReadMostly = 1, hipMemRangeAttributePreferredLocation = 2, hipMemRangeAttributeAccessedBy = 3,`
`hipMemRangeAttributeLastPrefetchLocation = 4,`
`hipMemRangeAttributeReadMostly = 1, hipMemRangeAttributePreferredLocation = 2, hipMemRangeAttributeAccessedBy = 3,`
`hipMemRangeAttributeLastPrefetchLocation = 4 }`

- enum **hipJitOption** {
hipJitOptionMaxRegisters = 0, **hipJitOptionThreadsPerBlock**, **hipJitOptionWallTime**, **hipJitOptionInfoLogBuffer**,
hipJitOptionInfoLogBufferSizeBytes, **hipJitOptionErrorLogBuffer**, **hipJitOptionErrorLogBufferSizeBytes**, **hipJitOptionOptimizationLevel**,
hipJitOptionTargetFromContext, **hipJitOptionTarget**, **hipJitOptionFallbackStrategy**, **hipJitOptionGenerateDebugInfo**,
hipJitOptionLogVerbose, **hipJitOptionGenerateLineInfo**, **hipJitOptionCacheMode**, **hipJitOptionSm3xOpt**,
hipJitOptionFastCompile, **hipJitOptionNumOptions**, **hipJitOptionMaxRegisters** = 0, **hipJitOptionThreadsPerBlock**,
hipJitOptionWallTime, **hipJitOptionInfoLogBuffer**, **hipJitOptionInfoLogBufferSizeBytes**, **hipJitOptionErrorLogBuffer**,
hipJitOptionErrorLogBufferSizeBytes, **hipJitOptionOptimizationLevel**, **hipJitOptionTargetFromContext**, **hipJitOptionTarget**,
hipJitOptionFallbackStrategy, **hipJitOptionGenerateDebugInfo**, **hipJitOptionLogVerbose**, **hipJitOptionGenerateLineInfo**,
hipJitOptionCacheMode, **hipJitOptionSm3xOpt**, **hipJitOptionFastCompile**, **hipJitOptionNumOptions** }
- enum **hipFuncAttribute** {
hipFuncAttributeMaxDynamicSharedMemorySize = 8, **hipFuncAttributePreferredSharedMemoryCarveout** = 9, **hipFuncAttributeMax**, **hipFuncAttributeMaxDynamicSharedMemorySize** = 8,
hipFuncAttributePreferredSharedMemoryCarveout = 9, **hipFuncAttributeMax** }
- enum **hipFuncCache_t** {
hipFuncCachePreferNone, **hipFuncCachePreferShared**, **hipFuncCachePreferL1**, **hipFuncCachePreferEqual**,
hipFuncCachePreferNone, **hipFuncCachePreferShared**, **hipFuncCachePreferL1**, **hipFuncCachePreferEqual**
}
- enum **hipSharedMemConfig** {
hipSharedMemBankSizeDefault, **hipSharedMemBankSizeFourByte**, **hipSharedMemBankSizeEightByte**,
hipSharedMemBankSizeDefault,
hipSharedMemBankSizeFourByte, **hipSharedMemBankSizeEightByte** }
- enum **hipExternalMemoryHandleType_enum** {
hipExternalMemoryHandleTypeOpaqueFd = 1, **hipExternalMemoryHandleTypeOpaqueWin32** = 2,
hipExternalMemoryHandleTypeOpaqueWin32Kmt = 3, **hipExternalMemoryHandleTypeD3D12Heap** = 4,
hipExternalMemoryHandleTypeD3D12Resource = 5, **hipExternalMemoryHandleTypeD3D11Resource** = 6, **hipExternalMemoryHandleTypeD3D11ResourceKmt** = 7, **hipExternalMemoryHandleTypeOpaqueFd** = 1,
hipExternalMemoryHandleTypeOpaqueWin32 = 2, **hipExternalMemoryHandleTypeOpaqueWin32Kmt** = 3, **hipExternalMemoryHandleTypeD3D12Heap** = 4, **hipExternalMemoryHandleTypeD3D12Resource** = 5,
hipExternalMemoryHandleTypeD3D11Resource = 6, **hipExternalMemoryHandleTypeD3D11ResourceKmt** = 7 }
- enum **hipDeviceAttribute_t** {
hipDeviceAttributeCudaCompatibleBegin = 0, **hipDeviceAttributeEccEnabled** = **hipDeviceAttributeCudaCompatibleBegin**, **hipDeviceAttributeAccessPolicyMaxWindowSize**, **hipDeviceAttributeAsyncEngineCount**,
hipDeviceAttributeCanMapHostMemory, **hipDeviceAttributeCanUseHostPointerForRegisteredMem**, **hipDeviceAttributeClockRate**, **hipDeviceAttributeComputeMode**,
hipDeviceAttributeComputePreemptionSupported, **hipDeviceAttributeConcurrentKernels**, **hipDeviceAttributeConcurrentManagedAccess**, **hipDeviceAttributeCooperativeLaunch**,
hipDeviceAttributeCooperativeMultiDeviceLaunch, **hipDeviceAttributeDeviceOverlap**, **hipDeviceAttributeDirectManagedMemoryAccess**, **hipDeviceAttributeGlobalL1CacheSupported**,
hipDeviceAttributeHostNativeAtomicSupported, **hipDeviceAttributeIntegrated**, **hipDeviceAttributeIsMultiGpuBoard**, **hipDeviceAttributeKernelExecTimeout**,
hipDeviceAttributeL2CacheSize, **hipDeviceAttributeLocalL1CacheSupported**, **hipDeviceAttributeLuid**,
hipDeviceAttributeLuidDeviceNodeMask,
hipDeviceAttributeComputeCapabilityMajor, **hipDeviceAttributeManagedMemory**, **hipDeviceAttributeMaxBlocksPerMultiProcessGroup**, **hipDeviceAttributeMaxBlockDimX**,
hipDeviceAttributeMaxBlockDimY, **hipDeviceAttributeMaxBlockDimZ**, **hipDeviceAttributeMaxGridDimX**,

```

hipDeviceAttributeMaxGridDimY,
hipDeviceAttributeMaxGridDimZ, hipDeviceAttributeMaxSurface1D, hipDeviceAttributeMaxSurface1DLayered,
hipDeviceAttributeMaxSurface2D,
hipDeviceAttributeMaxSurface2DLayered, hipDeviceAttributeMaxSurface3D, hipDeviceAttributeMaxSurfaceCubemap,
hipDeviceAttributeMaxSurfaceCubemapLayered,
hipDeviceAttributeMaxTexture1DWidth, hipDeviceAttributeMaxTexture1DLayered, hipDeviceAttributeMaxTexture1DLinear,
hipDeviceAttributeMaxTexture1DMipmap,
hipDeviceAttributeMaxTexture2DWidth, hipDeviceAttributeMaxTexture2DHeight, hipDeviceAttributeMaxTexture2DGather,
hipDeviceAttributeMaxTexture2DLayered,
hipDeviceAttributeMaxTexture2DLinear, hipDeviceAttributeMaxTexture2DMipmap, hipDeviceAttributeMaxTexture3DWidth,
hipDeviceAttributeMaxTexture3DHeight,
hipDeviceAttributeMaxTexture3DDepth, hipDeviceAttributeMaxTexture3DAlt, hipDeviceAttributeMaxTextureCubemap,
hipDeviceAttributeMaxTextureCubemapLayered,
hipDeviceAttributeMaxThreadsDim, hipDeviceAttributeMaxThreadsPerBlock, hipDeviceAttributeMaxThreadsPerMultiProcessor,
hipDeviceAttributeMaxPitch,
hipDeviceAttributeMemoryBusWidth, hipDeviceAttributeMemoryClockRate, hipDeviceAttributeComputeCapabilityMinor,
hipDeviceAttributeMultiGpuBoardGroupID,
hipDeviceAttributeMultiprocessorCount, hipDeviceAttributeName, hipDeviceAttributePageableMemoryAccess,
hipDeviceAttributePageableMemoryAccessUsesHostPageTables,
hipDeviceAttributePciBusId, hipDeviceAttributePciDeviceId, hipDeviceAttributePciDomainID, hipDeviceAttributePersistingL2Cache,
hipDeviceAttributeMaxRegistersPerBlock, hipDeviceAttributeMaxRegistersPerMultiprocessor, hipDeviceAttributeReservedSharedMemoryPerBlock,
hipDeviceAttributeMaxSharedMemoryPerBlock,
hipDeviceAttributeSharedMemPerBlockOptin, hipDeviceAttributeSharedMemPerMultiprocessor, hipDeviceAttributeSingleToDoublePrecisionFloatingPoint,
hipDeviceAttributeStreamPrioritiesSupported,
hipDeviceAttributeSurfaceAlignment, hipDeviceAttributeTccDriver, hipDeviceAttributeTextureAlignment,
hipDeviceAttributeTexturePitchAlignment,
hipDeviceAttributeTotalConstantMemory, hipDeviceAttributeTotalGlobalMem, hipDeviceAttributeUnifiedAddressing,
hipDeviceAttributeUuid,
hipDeviceAttributeWarpSize, hipDeviceAttributeCudaCompatibleEnd = 9999, hipDeviceAttributeAmdSpecificBegin = 10000, hipDeviceAttributeClockInstructionRate = hipDeviceAttributeAmdSpecificBegin,
hipDeviceAttributeArch, hipDeviceAttributeMaxSharedMemoryPerMultiprocessor, hipDeviceAttributeGcnArch,
hipDeviceAttributeGcnArchName,
hipDeviceAttributeHdpMemFlushCntl, hipDeviceAttributeHdpRegFlushCntl, hipDeviceAttributeCooperativeMultiDeviceUnmatchedGridDim,
hipDeviceAttributeCooperativeMultiDeviceUnmatchedGridDim,
hipDeviceAttributeCooperativeMultiDeviceUnmatchedBlockDim, hipDeviceAttributeCooperativeMultiDeviceUnmatchedSharedMemoryPerBlock,
hipDeviceAttributeIsLargeBar, hipDeviceAttributeAsicRevision,
hipDeviceAttributeCanUseStreamWaitValue, hipDeviceAttributeAmdSpecificEnd = 19999, hipDeviceAttributeVendorSpecificBegin = 20000 }
• enum hipComputeMode { hipComputeModeDefault = 0, hipComputeModeExclusive = 1, hipComputeModeProhibited = 2, hipComputeModeExclusiveProcess = 3 }

```

4.16.1 Detailed Description

4.16.2 Macro Definition Documentation

4.16.2.1 hipDeviceScheduleSpin [1/2]

```
#define hipDeviceScheduleSpin 0x1
```

may consume more power.

Dedicate a CPU core to spin-wait. Provides lowest latency, but burns a CPU core and

4.16.2.2 hipDeviceScheduleSpin [2/2]

```
#define hipDeviceScheduleSpin 0x1
```

may consume more power.

Dedicate a CPU core to spin-wait. Provides lowest latency, but burns a CPU core and

4.16.2.3 hipDeviceScheduleYield [1/2]

```
#define hipDeviceScheduleYield 0x2
```

power and is friendlier to other threads in the system.

Yield the CPU to the operating system when waiting. May increase latency, but lowers

4.16.2.4 hipDeviceScheduleYield [2/2]

```
#define hipDeviceScheduleYield 0x2
```

power and is friendlier to other threads in the system.

Yield the CPU to the operating system when waiting. May increase latency, but lowers

4.16.2.5 hipEventDefault [1/2]

```
#define hipEventDefault 0x0
```

Flags that can be used with hipEventCreateWithFlags:

Default flags

4.16.2.6 hipEventDefault [2/2]

```
#define hipEventDefault 0x0
```

Flags that can be used with hipEventCreateWithFlags:

Default flags

4.16.2.7 hipEventInterprocess [1/2]

```
#define hipEventInterprocess 0x4
```

Event can support IPC.

Warning

- not supported in HIP.

4.16.2.8 hipEventInterprocess [2/2]

```
#define hipEventInterprocess 0x4
```

Event can support IPC.

Warning

- not supported in HIP.

4.16.2.9 hipEventReleaseToSystem [1/2]

```
#define hipEventReleaseToSystem 0x80000000
```

< Use a device-scope release when recording this event. This flag is useful to obtain more precise timings of commands between events. The flag is a no-op on CUDA platforms.

4.16.2.10 hipEventReleaseToSystem [2/2]

```
#define hipEventReleaseToSystem 0x80000000
```

< Use a device-scope release when recording this event. This flag is useful to obtain more precise timings of commands between events. The flag is a no-op on CUDA platforms.

4.16.2.11 hipHostMallocCoherent [1/2]

```
#define hipHostMallocCoherent 0x40000000
```

allocation.

Allocate coherent memory. Overrides HIP_COHERENT_HOST_ALLOC for specific

4.16.2.12 hipHostMallocCoherent [2/2]

```
#define hipHostMallocCoherent 0x40000000
```

allocation.

Allocate coherent memory. Overrides HIP_COHERENT_HOST_ALLOC for specific

4.16.2.13 hipHostMallocDefault [1/2]

```
#define hipHostMallocDefault 0x0
```

Flags that can be used with hipHostMalloc.

< Use a system-scope release when recording this event. This flag is useful to make non-coherent host memory visible to the host. The flag is a no-op on CUDA platforms.

4.16.2.14 hipHostMallocDefault [2/2]

```
#define hipHostMallocDefault 0x0
```

Flags that can be used with hipHostMalloc.

< Use a system-scope release when recording this event. This flag is useful to make non-coherent host memory visible to the host. The flag is a no-op on CUDA platforms.

4.16.2.15 hipHostMallocMapped [1/2]

```
#define hipHostMallocMapped 0x2
```

can be obtained with [hipHostGetDevicePointer](#).

Map the allocation into the address space for the current device. The device pointer

4.16.2.16 hipHostMallocMapped [2/2]

```
#define hipHostMallocMapped 0x2
```

can be obtained with [hipHostGetDevicePointer](#).

Map the allocation into the address space for the current device. The device pointer

4.16.2.17 hipHostMallocNonCoherent [1/2]

```
#define hipHostMallocNonCoherent 0x80000000
```

allocation.

Allocate non-coherent memory. Overrides HIP_COHERENT_HOST_ALLOC for specific

4.16.2.18 hipHostMallocNonCoherent [2/2]

```
#define hipHostMallocNonCoherent 0x80000000
```

allocation.

Allocate non-coherent memory. Overrides HIP_COHERENT_HOST_ALLOC for specific

4.16.2.19 hipHostRegisterDefault [1/2]

```
#define hipHostRegisterDefault 0x0
```

Flags that can be used with hipHostRegister.

Memory is Mapped and Portable

4.16.2.20 hipHostRegisterDefault [2/2]

```
#define hipHostRegisterDefault 0x0
```

Flags that can be used with hipHostRegister.

Memory is Mapped and Portable

4.16.2.21 hipHostRegisterMapped [1/2]

```
#define hipHostRegisterMapped 0x2
```

can be obtained with [hipHostGetDevicePointer](#).

Map the allocation into the address space for the current device. The device pointer

4.16.2.22 hipHostRegisterMapped [2/2]

```
#define hipHostRegisterMapped 0x2
```

can be obtained with [hipHostGetDevicePointer](#).

Map the allocation into the address space for the current device. The device pointer

4.16.2.23 hipMemAttachSingle [1/2]

```
#define hipMemAttachSingle 0x04
```

the associated device

Memory can only be accessed by a single stream on

4.16.2.24 hipMemAttachSingle [2/2]

```
#define hipMemAttachSingle 0x04
```

the associated device

Memory can only be accessed by a single stream on

4.16.2.25 hipStreamDefault [1/2]

```
#define hipStreamDefault 0x00
```

Flags that can be used with [hipStreamCreateWithFlags](#).

Default stream creation flags. These are used with [hipStreamCreate\(\)](#).

4.16.2.26 hipStreamDefault [2/2]

```
#define hipStreamDefault 0x00
```

Flags that can be used with [hipStreamCreateWithFlags](#).

Default stream creation flags. These are used with [hipStreamCreate\(\)](#).

4.16.3 Typedef Documentation**4.16.3.1 dim3 [1/2]**

```
typedef struct dim3 dim3
```

Struct for data in 3D

4.16.3.2 dim3 [2/2]

```
typedef struct dim3 dim3
```

Struct for data in 3D

4.16.3.3 hipFuncAttribute [1/2]

```
typedef enum hipFuncAttribute hipFuncAttribute
```

Warning

On AMD devices and some Nvidia devices, these hints and controls are ignored.

4.16.3.4 hipFuncAttribute [2/2]

```
typedef enum hipFuncAttribute hipFuncAttribute
```

Warning

On AMD devices and some Nvidia devices, these hints and controls are ignored.

4.16.3.5 hipFuncCache_t [1/2]

```
typedef enum hipFuncCache_t hipFuncCache_t
```

Warning

On AMD devices and some Nvidia devices, these hints and controls are ignored.

4.16.3.6 hipFuncCache_t [2/2]

```
typedef enum hipFuncCache_t hipFuncCache_t
```

Warning

On AMD devices and some Nvidia devices, these hints and controls are ignored.

4.16.3.7 hipSharedMemConfig [1/2]

```
typedef enum hipSharedMemConfig hipSharedMemConfig
```

Warning

On AMD devices and some Nvidia devices, these hints and controls are ignored.

4.16.3.8 hipSharedMemConfig [2/2]

```
typedef enum hipSharedMemConfig hipSharedMemConfig
```

Warning

On AMD devices and some Nvidia devices, these hints and controls are ignored.

4.16.4 Enumeration Type Documentation**4.16.4.1 hipDeviceAttribute_t**

```
enum hipDeviceAttribute_t
```

Enumerator

hipDeviceAttributeEccEnabled	Whether ECC support is enabled.
hipDeviceAttributeAccessPolicyMaxWindowSize	Cuda only. The maximum size of the window policy in bytes.
hipDeviceAttributeAsyncEngineCount	Cuda only. Asynchronous engines number.
hipDeviceAttributeCanMapHostMemory	Whether host memory can be mapped into device address space.
hipDeviceAttributeCanUseHostPointerFor↔ RegisteredMem	Cuda only. Device can access host registered memory at the same virtual address as the CPU
hipDeviceAttributeClockRate	Peak clock frequency in kilohertz.
hipDeviceAttributeComputeMode	Compute mode that device is currently in.
hipDeviceAttributeComputePreemptionSupported	Cuda only. Device supports Compute Preemption.
hipDeviceAttributeConcurrentKernels	Device can possibly execute multiple kernels concurrently.

Enumerator

hipDeviceAttributeConcurrentManagedAccess	Device can coherently access managed memory concurrently with the CPU.
hipDeviceAttributeCooperativeLaunch	Support cooperative launch.
hipDeviceAttributeCooperativeMultiDeviceLaunch	Support cooperative launch on multiple devices.
hipDeviceAttributeDeviceOverlap	Cuda only. Device can concurrently copy memory and execute a kernel. Deprecated. Use instead <code>asyncEngineCount</code> .
hipDeviceAttributeDirectManagedMemAccessFrom↔ Host	Host can directly access managed memory on the device without migration
hipDeviceAttributeGlobalL1CacheSupported	Cuda only. Device supports caching globals in L1.
hipDeviceAttributeHostNativeAtomicSupported	Cuda only. Link between the device and the host supports native atomic operations.
hipDeviceAttributeIntegrated	Device is integrated GPU.
hipDeviceAttributeIsMultiGpuBoard	Multiple GPU devices.
hipDeviceAttributeKernelExecTimeout	Run time limit for kernels executed on the device.
hipDeviceAttributeL2CacheSize	Size of L2 cache in bytes. 0 if the device doesn't have L2 cache.
hipDeviceAttributeLocalL1CacheSupported	caching locals in L1 is supported
hipDeviceAttributeLuid	Cuda only. 8-byte locally unique identifier in 8 bytes. Undefined on TCC and non-Windows platforms.
hipDeviceAttributeLuidDeviceNodeMask	Cuda only. Luid device node mask. Undefined on TCC and non-Windows platforms.
hipDeviceAttributeComputeCapabilityMajor	Major compute capability version number.
hipDeviceAttributeManagedMemory	Device supports allocating managed memory on this system.
hipDeviceAttributeMaxBlocksPerMultiProcessor	Cuda only. Max block size per multiprocessor.
hipDeviceAttributeMaxBlockDimX	Max block size in width.
hipDeviceAttributeMaxBlockDimY	Max block size in height.
hipDeviceAttributeMaxBlockDimZ	Max block size in depth.
hipDeviceAttributeMaxGridDimX	Max grid size in width.
hipDeviceAttributeMaxGridDimY	Max grid size in height.
hipDeviceAttributeMaxGridDimZ	Max grid size in depth.
hipDeviceAttributeMaxSurface1D	Maximum size of 1D surface.
hipDeviceAttributeMaxSurface1DLayered	Cuda only. Maximum dimensions of 1D layered surface.
hipDeviceAttributeMaxSurface2D	Maximum dimension (width, height) of 2D surface.
hipDeviceAttributeMaxSurface2DLayered	Cuda only. Maximum dimensions of 2D layered surface.
hipDeviceAttributeMaxSurface3D	Maximum dimension (width, height, depth) of 3D surface.
hipDeviceAttributeMaxSurfaceCubemap	Cuda only. Maximum dimensions of Cubemap surface.
hipDeviceAttributeMaxSurfaceCubemapLayered	Cuda only. Maximum dimension of Cubemap layered surface.
hipDeviceAttributeMaxTexture1DWidth	Maximum size of 1D texture.
hipDeviceAttributeMaxTexture1DLayered	Cuda only. Maximum dimensions of 1D layered texture.
hipDeviceAttributeMaxTexture1DLinear	Maximum number of elements allocatable in a 1D linear texture. Use <code>cudaDeviceGetTexture1DLinearMaxWidth()</code> instead on Cuda.

Enumerator

hipDeviceAttributeMaxTexture1DMipmap	Cuda only. Maximum size of 1D mipmapped texture.
hipDeviceAttributeMaxTexture2DWidth	Maximum dimension width of 2D texture.
hipDeviceAttributeMaxTexture2DHeight	Maximum dimension height of 2D texture.
hipDeviceAttributeMaxTexture2DGather	Cuda only. Maximum dimensions of 2D texture if gather operations performed.
hipDeviceAttributeMaxTexture2DLayered	Cuda only. Maximum dimensions of 2D layered texture.
hipDeviceAttributeMaxTexture2DLinear	Cuda only. Maximum dimensions (width, height, pitch) of 2D textures bound to pitched memory.
hipDeviceAttributeMaxTexture2DMipmap	Cuda only. Maximum dimensions of 2D mipmapped texture.
hipDeviceAttributeMaxTexture3DWidth	Maximum dimension width of 3D texture.
hipDeviceAttributeMaxTexture3DHeight	Maximum dimension height of 3D texture.
hipDeviceAttributeMaxTexture3DDepth	Maximum dimension depth of 3D texture.
hipDeviceAttributeMaxTexture3DAlt	Cuda only. Maximum dimensions of alternate 3D texture.
hipDeviceAttributeMaxTextureCubemap	Cuda only. Maximum dimensions of Cubemap texture.
hipDeviceAttributeMaxTextureCubemapLayered	Cuda only. Maximum dimensions of Cubemap layered texture.
hipDeviceAttributeMaxThreadsDim	Maximum dimension of a block.
hipDeviceAttributeMaxThreadsPerBlock	Maximum number of threads per block.
hipDeviceAttributeMaxThreadsPerMultiProcessor	Maximum resident threads per multiprocessor.
hipDeviceAttributeMaxPitch	Maximum pitch in bytes allowed by memory copies.
hipDeviceAttributeMemoryBusWidth	Global memory bus width in bits.
hipDeviceAttributeMemoryClockRate	Peak memory clock frequency in kilohertz.
hipDeviceAttributeComputeCapabilityMinor	Minor compute capability version number.
hipDeviceAttributeMultiGpuBoardGroupID	Cuda only. Unique ID of device group on the same multi-GPU board.
hipDeviceAttributeMultiprocessorCount	Number of multiprocessors on the device.
hipDeviceAttributeName	Device name.
hipDeviceAttributePageableMemoryAccess	Device supports coherently accessing pageable memory without calling hipHostRegister on it
hipDeviceAttributePageableMemoryAccessUsesHostPageTables	Device accesses pageable memory via the host's page tables.
hipDeviceAttributePciBusId	PCI Bus ID.
hipDeviceAttributePciDeviceId	PCI Device ID.
hipDeviceAttributePciDomainId	PCI Domain ID.
hipDeviceAttributePersistingL2CacheMaxSize	Cuda11 only. Maximum l2 persisting lines capacity in bytes.
hipDeviceAttributeMaxRegistersPerBlock	32-bit registers available to a thread block. This number is shared by all thread blocks simultaneously resident on a multiprocessor.
hipDeviceAttributeMaxRegistersPerMultiprocessor	32-bit registers available per block.
hipDeviceAttributeReservedSharedMemPerBlock	Cuda11 only. Shared memory reserved by CUDA driver per block.
hipDeviceAttributeMaxSharedMemoryPerBlock	Maximum shared memory available per block in bytes.
hipDeviceAttributeSharedMemPerBlockOptIn	Cuda only. Maximum shared memory per block usable by special opt in.
hipDeviceAttributeSharedMemPerMultiprocessor	Cuda only. Shared memory available per multiprocessor.

Enumerator

hipDeviceAttributeSingleToDoublePrecisionPerfRatio	Cuda only. Performance ratio of single precision to double precision.
hipDeviceAttributeStreamPrioritiesSupported	Cuda only. Whether to support stream priorities.
hipDeviceAttributeSurfaceAlignment	Cuda only. Alignment requirement for surfaces.
hipDeviceAttributeTccDriver	Cuda only. Whether device is a Tesla device using TCC driver.
hipDeviceAttributeTextureAlignment	Alignment requirement for textures.
hipDeviceAttributeTexturePitchAlignment	Pitch alignment requirement for 2D texture references bound to pitched memory;.
hipDeviceAttributeTotalConstantMemory	Constant memory size in bytes.
hipDeviceAttributeTotalGlobalMem	Global memory available on device.
hipDeviceAttributeUnifiedAddressing	Cuda only. An unified address space shared with the host.
hipDeviceAttributeUuid	Cuda only. Unique ID in 16 byte.
hipDeviceAttributeWarpSize	Warp size in threads.
hipDeviceAttributeClockInstructionRate	Frequency in khz of the timer used by the device-side "clock*".
hipDeviceAttributeArch	Device architecture.
hipDeviceAttributeMaxSharedMemoryPerMultiprocessor	Maximum Shared Memory PerMultiprocessor.
hipDeviceAttributeGcnArch	Device gcn architecture.
hipDeviceAttributeGcnArchName	Device gcnArch name in 256 bytes.
hipDeviceAttributeHdpMemFlushCntl	Address of the HDP_MEM_COHERENCY_FLUSH_CNTL register.
hipDeviceAttributeHdpRegFlushCntl	Address of the HDP_REG_COHERENCY_FLUSH_CNTL register.
hipDeviceAttributeCooperativeMultiDeviceUnmatchedFunc	Supports cooperative launch on multiple devices with unmatched functions
hipDeviceAttributeCooperativeMultiDeviceUnmatchedGridDim	Supports cooperative launch on multiple devices with unmatched grid dimensions
hipDeviceAttributeCooperativeMultiDeviceUnmatchedBlockDim	Supports cooperative launch on multiple devices with unmatched block dimensions
hipDeviceAttributeCooperativeMultiDeviceUnmatchedSharedMem	Supports cooperative launch on multiple devices with unmatched shared memories
hipDeviceAttributeIlsLargeBar	Whether it is LargeBar.
hipDeviceAttributeAsicRevision	Revision of the GPU in this device.
hipDeviceAttributeCanUseStreamWaitValue	'1' if Device supports hipStreamWaitValue32() and hipStreamWaitValue64() , '0' otherwise.

4.16.4.2 hipFuncAttribute [1/2]

enum [hipFuncAttribute](#)

Warning

On AMD devices and some Nvidia devices, these hints and controls are ignored.

4.16.4.3 hipFuncAttribute [2/2]

enum [hipFuncAttribute](#)

Warning

On AMD devices and some Nvidia devices, these hints and controls are ignored.

4.16.4.4 hipFuncCache_t [1/2]

enum `hipFuncCache_t`

Warning

On AMD devices and some Nvidia devices, these hints and controls are ignored.

Enumerator

<code>hipFuncCachePreferNone</code>	no preference for shared memory or L1 (default)
<code>hipFuncCachePreferShared</code>	prefer larger shared memory and smaller L1 cache
<code>hipFuncCachePreferL1</code>	prefer larger L1 cache and smaller shared memory
<code>hipFuncCachePreferEqual</code>	prefer equal size L1 cache and shared memory
<code>hipFuncCachePreferNone</code>	no preference for shared memory or L1 (default)
<code>hipFuncCachePreferShared</code>	prefer larger shared memory and smaller L1 cache
<code>hipFuncCachePreferL1</code>	prefer larger L1 cache and smaller shared memory
<code>hipFuncCachePreferEqual</code>	prefer equal size L1 cache and shared memory

4.16.4.5 hipFuncCache_t [2/2]

enum `hipFuncCache_t`

Warning

On AMD devices and some Nvidia devices, these hints and controls are ignored.

Enumerator

<code>hipFuncCachePreferNone</code>	no preference for shared memory or L1 (default)
<code>hipFuncCachePreferShared</code>	prefer larger shared memory and smaller L1 cache
<code>hipFuncCachePreferL1</code>	prefer larger L1 cache and smaller shared memory
<code>hipFuncCachePreferEqual</code>	prefer equal size L1 cache and shared memory
<code>hipFuncCachePreferNone</code>	no preference for shared memory or L1 (default)
<code>hipFuncCachePreferShared</code>	prefer larger shared memory and smaller L1 cache
<code>hipFuncCachePreferL1</code>	prefer larger L1 cache and smaller shared memory
<code>hipFuncCachePreferEqual</code>	prefer equal size L1 cache and shared memory

4.16.4.6 hipMemoryAdvise [1/2]

enum `hipMemoryAdvise`

Enumerator

<code>hipMemAdviseSetReadMostly</code>	Data will mostly be read and only occasionally be written to
<code>hipMemAdviseUnsetReadMostly</code>	Undo the effect of <code>hipMemAdviseSetReadMostly</code> .

Enumerator

hipMemAdviseSetPreferredLocation	Set the preferred location for the data as the specified device
hipMemAdviseUnsetPreferredLocation	Clear the preferred location for the data.
hipMemAdviseSetAccessedBy	Data will be accessed by the specified device, so prevent page faults as much as possible
hipMemAdviseUnsetAccessedBy	Let the Unified Memory subsystem decide on the page faulting policy for the specified device
hipMemAdviseSetReadMostly	Data will mostly be read and only occasionally be written to
hipMemAdviseUnsetReadMostly	Undo the effect of hipMemAdviseSetReadMostly.
hipMemAdviseSetPreferredLocation	Set the preferred location for the data as the specified device
hipMemAdviseUnsetPreferredLocation	Clear the preferred location for the data.
hipMemAdviseSetAccessedBy	Data will be accessed by the specified device, so prevent page faults as much as possible
hipMemAdviseUnsetAccessedBy	Let the Unified Memory subsystem decide on the page faulting policy for the specified device

4.16.4.7 hipMemoryAdvise [2/2]

enum `hipMemoryAdvise`

Enumerator

hipMemAdviseSetReadMostly	Data will mostly be read and only occasionally be written to
hipMemAdviseUnsetReadMostly	Undo the effect of hipMemAdviseSetReadMostly.
hipMemAdviseSetPreferredLocation	Set the preferred location for the data as the specified device
hipMemAdviseUnsetPreferredLocation	Clear the preferred location for the data.
hipMemAdviseSetAccessedBy	Data will be accessed by the specified device, so prevent page faults as much as possible
hipMemAdviseUnsetAccessedBy	Let the Unified Memory subsystem decide on the page faulting policy for the specified device
hipMemAdviseSetReadMostly	Data will mostly be read and only occasionally be written to
hipMemAdviseUnsetReadMostly	Undo the effect of hipMemAdviseSetReadMostly.
hipMemAdviseSetPreferredLocation	Set the preferred location for the data as the specified device
hipMemAdviseUnsetPreferredLocation	Clear the preferred location for the data.
hipMemAdviseSetAccessedBy	Data will be accessed by the specified device, so prevent page faults as much as possible
hipMemAdviseUnsetAccessedBy	Let the Unified Memory subsystem decide on the page faulting policy for the specified device

4.16.4.8 hipMemRangeAttribute [1/2]

enum `hipMemRangeAttribute`

Enumerator

hipMemRangeAttributeReadMostly	Whether the range will mostly be read and only occasionally be written to
hipMemRangeAttributePreferredLocation	The preferred location of the range.

Enumerator

hipMemRangeAttributeAccessedBy	Memory range has cudaMemAdviseSetAccessedBy set for specified device
hipMemRangeAttributeLastPrefetchLocation	The last location to which the range was prefetched.
hipMemRangeAttributeReadMostly	Whether the range will mostly be read and only occasionally be written to
hipMemRangeAttributePreferredLocation	The preferred location of the range.
hipMemRangeAttributeAccessedBy	Memory range has cudaMemAdviseSetAccessedBy set for specified device
hipMemRangeAttributeLastPrefetchLocation	The last location to which the range was prefetched.

4.16.4.9 hipMemRangeAttribute [2/2]

enum `hipMemRangeAttribute`

Enumerator

hipMemRangeAttributeReadMostly	Whether the range will mostly be read and only occasionally be written to
hipMemRangeAttributePreferredLocation	The preferred location of the range.
hipMemRangeAttributeAccessedBy	Memory range has cudaMemAdviseSetAccessedBy set for specified device
hipMemRangeAttributeLastPrefetchLocation	The last location to which the range was prefetched.
hipMemRangeAttributeReadMostly	Whether the range will mostly be read and only occasionally be written to
hipMemRangeAttributePreferredLocation	The preferred location of the range.
hipMemRangeAttributeAccessedBy	Memory range has cudaMemAdviseSetAccessedBy set for specified device
hipMemRangeAttributeLastPrefetchLocation	The last location to which the range was prefetched.

4.16.4.10 hipSharedMemConfig [1/2]

enum `hipSharedMemConfig`

Warning

On AMD devices and some Nvidia devices, these hints and controls are ignored.

Enumerator

hipSharedMemBankSizeDefault	The compiler selects a device-specific value for the banking.
hipSharedMemBankSizeFourByte	Shared mem is banked at 4-bytes intervals and performs best when adjacent threads access data 4 bytes apart.
hipSharedMemBankSizeEightByte	Shared mem is banked at 8-byte intervals and performs best when adjacent threads access data 4 bytes apart.
hipSharedMemBankSizeDefault	The compiler selects a device-specific value for the banking.
hipSharedMemBankSizeFourByte	Shared mem is banked at 4-bytes intervals and performs best when adjacent threads access data 4 bytes apart.
hipSharedMemBankSizeEightByte	Shared mem is banked at 8-byte intervals and performs best when adjacent threads access data 4 bytes apart.

4.16.4.11 hipSharedMemConfig [2/2]

enum [hipSharedMemConfig](#)

Warning

On AMD devices and some Nvidia devices, these hints and controls are ignored.

Enumerator

hipSharedMemBankSizeDefault	The compiler selects a device-specific value for the banking.
hipSharedMemBankSizeFourByte	Shared mem is banked at 4-bytes intervals and performs best when adjacent threads access data 4 bytes apart.
hipSharedMemBankSizeEightByte	Shared mem is banked at 8-byte intervals and performs best when adjacent threads access data 4 bytes apart.
hipSharedMemBankSizeDefault	The compiler selects a device-specific value for the banking.
hipSharedMemBankSizeFourByte	Shared mem is banked at 4-bytes intervals and performs best when adjacent threads access data 4 bytes apart.
hipSharedMemBankSizeEightByte	Shared mem is banked at 8-byte intervals and performs best when adjacent threads access data 4 bytes apart.

4.17 Managed Memory (ROCm HMM)

Functions

- `hipError_t hipMallocManaged` (void **dev_ptr, size_t size, unsigned int flags __dparm(hipMemAttachGlobal))
Allocates memory that will be automatically managed by AMD HMM.
- `hipError_t hipMemPrefetchAsync` (const void *dev_ptr, size_t count, int device, hipStream_t stream __↔
dparm(0))
Prefetches memory to the specified destination device using AMD HMM.
- `hipError_t hipMemAdvise` (const void *dev_ptr, size_t count, hipMemoryAdvise advice, int device)
Advise about the usage of a given memory range to AMD HMM.
- `hipError_t hipMemRangeGetAttribute` (void *data, size_t data_size, hipMemRangeAttribute attribute, const
void *dev_ptr, size_t count)
Query an attribute of a given memory range in AMD HMM.
- `hipError_t hipMemRangeGetAttributes` (void **data, size_t *data_sizes, hipMemRangeAttribute *attributes,
size_t num_attributes, const void *dev_ptr, size_t count)
Query attributes of a given memory range in AMD HMM.
- `hipError_t hipStreamAttachMemAsync` (hipStream_t stream, hipDeviceptr_t *dev_ptr, size_t length __↔
dparm(0), unsigned int flags __dparm(hipMemAttachSingle))
Attach memory to a stream asynchronously in AMD HMM.

4.17.1 Detailed Description

This section describes the managed memory management functions of HIP runtime API.

4.17.2 Function Documentation

4.17.2.1 hipMallocManaged()

```
hipError_t hipMallocManaged (
    void ** dev_ptr,
    size_t size,
    unsigned int flags __dparm(hipMemAttachGlobal )
```

Allocates memory that will be automatically managed by AMD HMM.

Parameters

out	<i>dev_ptr</i>	- pointer to allocated device memory
in	<i>size</i>	- requested allocation size in bytes
in	<i>flags</i>	- must be either hipMemAttachGlobal or hipMemAttachHost (defaults to hipMemAttachGlobal)

Returns

#hipSuccess, #hipErrorMemoryAllocation, #hipErrorNotSupported, #hipErrorInvalidValue

4.17.2.2 hipMemAdvise()

```
hipError_t hipMemAdvise (
    const void * dev_ptr,
    size_t count,
    hipMemoryAdvise advice,
    int device )
```

Advise about the usage of a given memory range to AMD HMM.

Parameters

in	<i>dev_ptr</i>	pointer to memory to set the advice for
in	<i>count</i>	size in bytes of the memory range
in	<i>advice</i>	advice to be applied for the specified memory range
in	<i>device</i>	device to apply the advice for

Returns

#hipSuccess, #hipErrorInvalidValue

4.17.2.3 hipMemPrefetchAsync()

```
hipError_t hipMemPrefetchAsync (
    const void * dev_ptr,
    size_t count,
    int device,
    hipStream_t stream __dparm0 )
```

Prefetches memory to the specified destination device using AMD HMM.

Parameters

in	<i>dev_ptr</i>	pointer to be prefetched
in	<i>count</i>	size in bytes for prefetching
in	<i>device</i>	destination device to prefetch to
in	<i>stream</i>	stream to enqueue prefetch operation

Returns

#hipSuccess, #hipErrorInvalidValue

4.17.2.4 hipMemRangeGetAttribute()

```
hipError_t hipMemRangeGetAttribute (
    void * data,
    size_t data_size,
    hipMemRangeAttribute attribute,
    const void * dev_ptr,
    size_t count )
```

Query an attribute of a given memory range in AMD HMM.

Parameters

	<i>[in/out]</i>	data a pointer to a memory location where the result of each attribute query will be written to
in	<i>data_size</i>	the size of data
in	<i>attribute</i>	the attribute to query
in	<i>dev_ptr</i>	start of the range to query
in	<i>count</i>	size of the range to query

Returns

#hipSuccess, #hipErrorInvalidValue

4.17.2.5 hipMemRangeGetAttributes()

```
hipError_t hipMemRangeGetAttributes (
    void ** data,
    size_t * data_sizes,
    hipMemRangeAttribute * attributes,
    size_t num_attributes,
    const void * dev_ptr,
    size_t count )
```

Query attributes of a given memory range in AMD HMM.

Parameters

	<i>[in/out]</i>	data a two-dimensional array containing pointers to memory locations where the result of each attribute query will be written to
in	<i>data_sizes</i>	an array, containing the sizes of each result
in	<i>attributes</i>	the attribute to query
in	<i>num_attributes</i>	an array of attributes to query (numAttributes and the number of attributes in this array should match)
in	<i>dev_ptr</i>	start of the range to query
in	<i>count</i>	size of the range to query

Returns

#hipSuccess, #hipErrorInvalidValue

4.17.2.6 hipStreamAttachMemAsync()

```
hipError_t hipStreamAttachMemAsync (
    hipStream_t stream,
    hipDeviceptr_t * dev_ptr,
    size_t length __dparm0,
    unsigned int flags __dparmhipMemAttachSingle )
```

Attach memory to a stream asynchronously in AMD HMM.

Parameters

in	<i>stream</i>	- stream in which to enqueue the attach operation
in	<i>dev_ptr</i>	- pointer to memory (must be a pointer to managed memory or to a valid host-accessible region of system-allocated memory)
in	<i>length</i>	- length of memory (defaults to zero)
in	<i>flags</i>	- must be one of cudaMemAttachGlobal, cudaMemAttachHost or cudaMemAttachSingle (defaults to cudaMemAttachSingle)

Returns

#hipSuccess, #hipErrorInvalidValue

4.18 Context Management [Deprecated]

Functions

- hipError_t [hipCtxCreate](#) (hipCtx_t *ctx, unsigned int flags, hipDevice_t device)
Create a context and set it as current/ default context.
- hipError_t [hipCtxDestroy](#) (hipCtx_t ctx)
Destroy a HIP context.
- hipError_t [hipCtxPopCurrent](#) (hipCtx_t *ctx)
Pop the current/default context and return the popped context.
- hipError_t [hipCtxPushCurrent](#) (hipCtx_t ctx)
Push the context to be set as current/ default context.
- hipError_t [hipCtxSetCurrent](#) (hipCtx_t ctx)
Set the passed context as current/default.
- hipError_t [hipCtxGetCurrent](#) (hipCtx_t *ctx)
Get the handle of the current/ default context.
- hipError_t [hipCtxGetDevice](#) (hipDevice_t *device)
Get the handle of the device associated with current/default context.
- hipError_t [hipCtxGetApiVersion](#) (hipCtx_t ctx, int *apiVersion)
Returns the approximate HIP api version.
- hipError_t [hipCtxGetCacheConfig](#) (hipFuncCache_t *cacheConfig)
Set Cache configuration for a specific function.
- hipError_t [hipCtxSetCacheConfig](#) (hipFuncCache_t cacheConfig)
Set L1/Shared cache partition.
- hipError_t [hipCtxSetSharedMemConfig](#) (hipSharedMemConfig config)
Set Shared memory bank configuration.
- hipError_t [hipCtxGetSharedMemConfig](#) (hipSharedMemConfig *pConfig)
Get Shared memory bank configuration.
- hipError_t [hipCtxSynchronize](#) (void)
Blocks until the default context has completed all preceding requested tasks.
- hipError_t [hipCtxGetFlags](#) (unsigned int *flags)
Return flags used for creating default context.
- hipError_t [hipCtxEnablePeerAccess](#) (hipCtx_t peerCtx, unsigned int flags)
Enables direct access to memory allocations in a peer context.
- hipError_t [hipCtxDisablePeerAccess](#) (hipCtx_t peerCtx)
Disable direct access from current context's virtual address space to memory allocations physically located on a peer context. Disables direct access to memory allocations in a peer context and unregisters any registered allocations.

4.18.1 Detailed Description

This section describes the deprecated context management functions of HIP runtime API.

4.18.2 Function Documentation

4.18.2.1 hipCtxCreate()

```
hipError_t hipCtxCreate (
    hipCtx_t * ctx,
    unsigned int flags,
    hipDevice_t device )
```

Create a context and set it as current/ default context.

Parameters

out	<i>ctx</i>	
in	<i>flags</i>	
in	<i>associated</i>	device handle

Returns

#hipSuccess

See also

[hipCtxDestroy](#), [hipCtxGetFlags](#), [hipCtxPopCurrent](#), [hipCtxGetCurrent](#), [hipCtxPushCurrent](#), [hipCtxSetCacheConfig](#), [hipCtxSynchronize](#), [hipCtxGetDevice](#)

4.18.2.2 hipCtxDestroy()

```
hipError_t hipCtxDestroy (
    hipCtx_t ctx )
```

Destroy a HIP context.

Parameters

in	<i>ctx</i>	Context to destroy
----	------------	--------------------

Returns

#hipSuccess, #hipErrorInvalidValue

See also

[hipCtxCreate](#), [hipCtxGetFlags](#), [hipCtxPopCurrent](#), [hipCtxGetCurrent](#), [hipCtxSetCurrent](#), [hipCtxPushCurrent](#), [hipCtxSetCacheConfig](#), [hipCtxSynchronize](#), [hipCtxGetDevice](#)

4.18.2.3 hipCtxDisablePeerAccess()

```
hipError_t hipCtxDisablePeerAccess (
    hipCtx_t peerCtx )
```

Disable direct access from current context's virtual address space to memory allocations physically located on a peer context. Disables direct access to memory allocations in a peer context and unregisters any registered allocations.

Returns hipErrorPeerAccessNotEnabled if direct access to memory on peerDevice has not yet been enabled from the current device.

Parameters

in	<i>peerCtx</i>	
----	----------------	--

Returns

#hipSuccess, #hipErrorPeerAccessNotEnabled

See also

[hipCtxCreate](#), [hipCtxDestroy](#), [hipCtxGetFlags](#), [hipCtxPopCurrent](#), [hipCtxGetCurrent](#), [hipCtxSetCurrent](#), [hipCtxPushCurrent](#), [hipCtxSetCacheConfig](#), [hipCtxSynchronize](#), [hipCtxGetDevice](#)

Warning

PeerToPeer support is experimental.

4.18.2.4 hipCtxEnablePeerAccess()

```
hipError_t hipCtxEnablePeerAccess (
    hipCtx_t peerCtx,
    unsigned int flags )
```

Enables direct access to memory allocations in a peer context.

Memory which already allocated on peer device will be mapped into the address space of the current device. In addition, all future memory allocations on peerDeviceId will be mapped into the address space of the current device when the memory is allocated. The peer memory remains accessible from the current device until a call to `hipDeviceDisablePeerAccess` or `hipDeviceReset`.

Parameters

in	<i>peerCtx</i>	
in	<i>flags</i>	

Returns

`#hipSuccess`, `#hipErrorInvalidDevice`, `#hipErrorInvalidValue`, `#hipErrorPeerAccessAlreadyEnabled`

See also

[hipCtxCreate](#), [hipCtxDestroy](#), [hipCtxGetFlags](#), [hipCtxPopCurrent](#), [hipCtxGetCurrent](#), [hipCtxSetCurrent](#), [hipCtxPushCurrent](#), [hipCtxSetCacheConfig](#), [hipCtxSynchronize](#), [hipCtxGetDevice](#)

Warning

PeerToPeer support is experimental.

4.18.2.5 hipCtxGetApiVersion()

```
hipError_t hipCtxGetApiVersion (
    hipCtx_t ctx,
    int * apiVersion )
```

Returns the approximate HIP api version.

Parameters

in	<i>ctx</i>	Context to check
out	<i>apiVersion</i>	

Returns

`#hipSuccess`

Warning

The HIP feature set does not correspond to an exact CUDA SDK api revision. This function always set `*apiVersion` to 4 as an approximation though HIP supports some features which were introduced in later CUDA SDK revisions. HIP apps code should not rely on the api revision number here and should use arch feature flags to test device capabilities or conditional compilation.

See also

[hipCtxCreate](#), [hipCtxDestroy](#), [hipCtxGetDevice](#), [hipCtxGetFlags](#), [hipCtxPopCurrent](#), [hipCtxPushCurrent](#), [hipCtxSetCacheConfig](#), [hipCtxSynchronize](#), [hipCtxGetDevice](#)

4.18.2.6 hipCtxGetCacheConfig()

```
hipError_t hipCtxGetCacheConfig (
    hipFuncCache_t * cacheConfig )
```

Set Cache configuration for a specific function.

Parameters

out	<i>cacheConfiguration</i>	
-----	---------------------------	--

Returns

`#hipSuccess`

Warning

AMD devices and some Nvidia GPUS do not support reconfigurable cache. This hint is ignored on those architectures.

See also

[hipCtxCreate](#), [hipCtxDestroy](#), [hipCtxGetFlags](#), [hipCtxPopCurrent](#), [hipCtxGetCurrent](#), [hipCtxSetCurrent](#), [hipCtxPushCurrent](#), [hipCtxSetCacheConfig](#), [hipCtxSynchronize](#), [hipCtxGetDevice](#)

4.18.2.7 hipCtxGetCurrent()

```
hipError_t hipCtxGetCurrent (
    hipCtx_t * ctx )
```

Get the handle of the current/ default context.

Parameters

out	<i>ctx</i>	
-----	------------	--

Returns

`#hipSuccess`, `#hipErrorInvalidContext`

See also

[hipCtxCreate](#), [hipCtxDestroy](#), [hipCtxGetDevice](#), [hipCtxGetFlags](#), [hipCtxPopCurrent](#), [hipCtxPushCurrent](#), [hipCtxSetCacheConfig](#), [hipCtxSynchronize](#), [hipCtxGetDevice](#)

4.18.2.8 hipCtxGetDevice()

```
hipError_t hipCtxGetDevice (
    hipDevice_t * device )
```

Get the handle of the device associated with current/default context.

Parameters

out	<i>device</i>	
-----	---------------	--

Returns

#hipSuccess, #hipErrorInvalidContext

See also

[hipCtxCreate](#), [hipCtxDestroy](#), [hipCtxGetFlags](#), [hipCtxPopCurrent](#), [hipCtxGetCurrent](#), [hipCtxPushCurrent](#), [hipCtxSetCacheConfig](#), [hipCtxSynchronize](#)

4.18.2.9 hipCtxGetFlags()

```
hipError_t hipCtxGetFlags (
    unsigned int * flags )
```

Return flags used for creating default context.

Parameters

out	<i>flags</i>	
-----	--------------	--

Returns

#hipSuccess

See also

[hipCtxCreate](#), [hipCtxDestroy](#), [hipCtxPopCurrent](#), [hipCtxGetCurrent](#), [hipCtxGetCurrent](#), [hipCtxSetCurrent](#), [hipCtxPushCurrent](#), [hipCtxSetCacheConfig](#), [hipCtxSynchronize](#), [hipCtxGetDevice](#)

4.18.2.10 hipCtxGetSharedMemConfig()

```
hipError_t hipCtxGetSharedMemConfig (
    hipSharedMemConfig * pConfig )
```

Get Shared memory bank configuration.

Parameters

out	<i>sharedMemoryConfiguration</i>	
-----	----------------------------------	--

Returns

#hipSuccess

Warning

AMD devices and some Nvidia GPUS do not support shared cache banking, and the hint is ignored on those architectures.

See also

[hipCtxCreate](#), [hipCtxDestroy](#), [hipCtxGetFlags](#), [hipCtxPopCurrent](#), [hipCtxGetCurrent](#), [hipCtxSetCurrent](#), [hipCtxPushCurrent](#), [hipCtxSetCacheConfig](#), [hipCtxSynchronize](#), [hipCtxGetDevice](#)

4.18.2.11 hipCtxPopCurrent()

```
hipError_t hipCtxPopCurrent (
    hipCtx_t * ctx )
```

Pop the current/default context and return the popped context.

Parameters

out	<i>ctx</i>	
-----	------------	--

Returns

#hipSuccess, #hipErrorInvalidContext

See also

[hipCtxCreate](#), [hipCtxDestroy](#), [hipCtxGetFlags](#), [hipCtxSetCurrent](#), [hipCtxGetCurrent](#), [hipCtxPushCurrent](#), [hipCtxSetCacheConfig](#), [hipCtxSynchronize](#), [hipCtxGetDevice](#)

4.18.2.12 hipCtxPushCurrent()

```
hipError_t hipCtxPushCurrent (
    hipCtx_t ctx )
```

Push the context to be set as current/ default context.

Parameters

in	<i>ctx</i>	
----	------------	--

Returns

#hipSuccess, #hipErrorInvalidContext

See also

[hipCtxCreate](#), [hipCtxDestroy](#), [hipCtxGetFlags](#), [hipCtxPopCurrent](#), [hipCtxGetCurrent](#), [hipCtxPushCurrent](#), [hipCtxSetCacheConfig](#), [hipCtxSynchronize](#), [hipCtxGetDevice](#)

4.18.2.13 hipCtxSetCacheConfig()

```
hipError_t hipCtxSetCacheConfig (
    hipFuncCache_t cacheConfig )
```

Set L1/Shared cache partition.

Parameters

in	<i>cacheConfiguration</i>	
----	---------------------------	--

Returns

#hipSuccess

Warning

AMD devices and some Nvidia GPUS do not support reconfigurable cache. This hint is ignored on those architectures.

See also

[hipCtxCreate](#), [hipCtxDestroy](#), [hipCtxGetFlags](#), [hipCtxPopCurrent](#), [hipCtxGetCurrent](#), [hipCtxSetCurrent](#), [hipCtxPushCurrent](#), [hipCtxSetCacheConfig](#), [hipCtxSynchronize](#), [hipCtxGetDevice](#)

4.18.2.14 hipCtxSetCurrent()

```
hipError_t hipCtxSetCurrent (
    hipCtx_t ctx )
```

Set the passed context as current/default.

Parameters

in	<i>ctx</i>	
----	------------	--

Returns

#hipSuccess, #hipErrorInvalidContext

See also

[hipCtxCreate](#), [hipCtxDestroy](#), [hipCtxGetFlags](#), [hipCtxPopCurrent](#), [hipCtxGetCurrent](#), [hipCtxPushCurrent](#), [hipCtxSetCacheConfig](#), [hipCtxSynchronize](#), [hipCtxGetDevice](#)

4.18.2.15 hipCtxSetSharedMemConfig()

```
hipError_t hipCtxSetSharedMemConfig (
    hipSharedMemConfig config )
```

Set Shared memory bank configuration.

Parameters

in	<i>sharedMemoryConfiguration</i>	
----	----------------------------------	--

Returns

#hipSuccess

Warning

AMD devices and some Nvidia GPUS do not support shared cache banking, and the hint is ignored on those architectures.

See also

[hipCtxCreate](#), [hipCtxDestroy](#), [hipCtxGetFlags](#), [hipCtxPopCurrent](#), [hipCtxGetCurrent](#), [hipCtxSetCurrent](#), [hipCtxPushCurrent](#), [hipCtxSetCacheConfig](#), [hipCtxSynchronize](#), [hipCtxGetDevice](#)

4.18.2.16 hipCtxSynchronize()

```
hipError_t hipCtxSynchronize (
    void )
```

Blocks until the default context has completed all preceding requested tasks.

Returns

#hipSuccess

Warning

This function waits for all streams on the default context to complete execution, and then returns.

See also

[hipCtxCreate](#), [hipCtxDestroy](#), [hipCtxGetFlags](#), [hipCtxPopCurrent](#), [hipCtxGetCurrent](#), [hipCtxSetCurrent](#), [hipCtxPushCurrent](#), [hipCtxSetCacheConfig](#), [hipCtxGetDevice](#)

4.19 Texture Management [Deprecated]

Functions

- `hipError_t hipBindTexture` (`size_t *offset`, `const textureReference *tex`, `const void *devPtr`, `const hipChannelFormatDesc *desc`, `size_t size __dparm(UINT_MAX)`)
- `hipError_t hipBindTexture2D` (`size_t *offset`, `const textureReference *tex`, `const void *devPtr`, `const hipChannelFormatDesc *desc`, `size_t width`, `size_t height`, `size_t pitch`)
- `hipError_t hipBindTextureToArray` (`const textureReference *tex`, `hipArray_const_t array`, `const hipChannelFormatDesc *desc`)
- `hipError_t hipGetTextureAlignmentOffset` (`size_t *offset`, `const textureReference *texref`)
- `hipError_t hipUnbindTexture` (`const textureReference *tex`)

4.19.1 Detailed Description

This section describes the deprecated texture management functions of HIP runtime API.

Chapter 5

Class Documentation

5.1 `__half2_raw` Struct Reference

Public Attributes

- unsigned short `x`
- unsigned short `y`

5.2 `__half_raw` Struct Reference

Public Attributes

- unsigned short `x`

5.3 `__hip_enable_if< __B, __T >` Struct Template Reference

5.4 `__hip_enable_if< true, __T >` Struct Template Reference

Public Types

- typedef `__T` `type`
- typedef `__T` `type`

5.5 `char1` Union Reference

Public Attributes

- char `data`

5.6 `char16` Union Reference

Public Attributes

- char `data` [16]

5.7 `char2` Union Reference

Public Attributes

- char `data` [2]

5.8 char3 Union Reference

Public Attributes

- [char4](#) `data`

5.9 char4 Union Reference

Public Attributes

- `char data` [4]

5.10 char8 Union Reference

Public Attributes

- `char data` [8]

5.11 dim3 Struct Reference

Public Attributes

- [uint32_t](#) `x`
`x`
- [uint32_t](#) `y`
`y`
- [uint32_t](#) `z`
`z`

5.11.1 Detailed Description

Struct for data in 3D

5.12 double1 Union Reference

Public Attributes

- `double data`

5.13 double16 Union Reference

Public Attributes

- `double data` [16]

5.14 double2 Union Reference

Public Attributes

- `double data` [2]

5.15 double3 Union Reference

Public Attributes

- [double4](#) data

5.16 double4 Union Reference

Public Attributes

- double data [4]

5.17 double8 Union Reference

Public Attributes

- double data [8]

5.18 float1 Union Reference

Public Attributes

- float data

5.19 float16 Union Reference

Public Attributes

- float data [16]

5.20 float2 Union Reference

Public Attributes

- float data [2]

5.21 float3 Union Reference

Public Attributes

- [float4](#) data

5.22 float4 Union Reference

Public Attributes

- float data [4]

5.23 float8 Union Reference

Public Attributes

- float data [8]

5.24 gl_dim3 Struct Reference

Public Member Functions

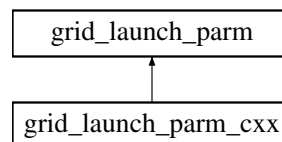
- **gl_dim3** (uint32_t _x=1, uint32_t _y=1, uint32_t _z=1)
- **gl_dim3** (uint32_t _x=1, uint32_t _y=1, uint32_t _z=1)

Public Attributes

- int **x**
- int **y**
- int **z**

5.25 grid_launch_parm Struct Reference

Inheritance diagram for grid_launch_parm:



Public Attributes

- [gl_dim3 grid_dim](#)
Grid dimensions.
- [gl_dim3 group_dim](#)
Group dimensions.
- unsigned int [dynamic_group_mem_bytes](#)
- enum gl_barrier_bit [barrier_bit](#)
- unsigned int [launch_fence](#)
- hc::accelerator_view * [av](#)
- hc::completion_future * [cf](#)

5.25.1 Member Data Documentation

5.25.1.1 av

hc::accelerator_view * grid_launch_parm::av

Pointer to the accelerator_view where the kernel should execute. If NULL, the default view on the default accelerator is used.

5.25.1.2 barrier_bit

enum gl_barrier_bit grid_launch_parm::barrier_bit

Control setting of barrier bit on per-packet basis: See gl_barrier_bit description. Placeholder, is not used to control packet dispatch yet

5.25.1.3 cf

hc::completion_future * grid_launch_parm::cf

Pointer to the completion_future used to track the status of the command. If NULL, the command does not write status. In this case, synchronization can be enforced with queue-level waits or waiting on younger commands.

5.25.1.4 dynamic_group_mem_bytes

`unsigned int grid_launch_parm::dynamic_group_mem_bytes`

Amount of dynamic group memory to use with the kernel launch. This memory is in addition to the amount used statically in the kernel.

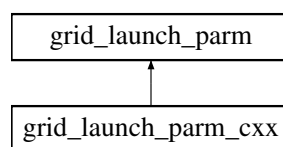
5.25.1.5 launch_fence

`unsigned int grid_launch_parm::launch_fence`

Value of packet fences to apply to launch. The correspond to the value of bits 9:14 in the AQL packet, see `HSA_PACKET_HEADER_ACQUIRE_FENCE_SCOPE` and `hsa_fence_scope_t`.

5.26 grid_launch_parm_cxx Class Reference

Inheritance diagram for `grid_launch_parm_cxx`:



Public Member Functions

- `__attribute__((annotate("serialize"))) void __cxxamp_serialize(Kalmar`
- `__attribute__((annotate("serialize"))) void __cxxamp_serialize(Kalmar`

Additional Inherited Members

5.27 HIP_ARRAY3D_DESCRIPTOR Struct Reference

Public Attributes

- `size_t Width`
- `size_t Height`
- `size_t Depth`
- `enum hipArray_Format Format`
- `unsigned int NumChannels`
- `unsigned int Flags`

5.28 HIP_ARRAY_DESCRIPTOR Struct Reference

Public Attributes

- `size_t Width`
- `size_t Height`
- `enum hipArray_Format Format`
- `unsigned int NumChannels`

5.29 hip_bfloat16 Struct Reference

Struct to represent a 16 bit brain floating point number.

Public Attributes

- `uint16_t data`

5.29.1 Detailed Description

Struct to represent a 16 bit brain floating point number.

5.30 hip_Memcpy2D Struct Reference

Public Attributes

- `size_t` **srcXInBytes**
- `size_t` **srcY**
- `hipMemoryType` **srcMemoryType**
- `const void *` **srcHost**
- `hipDeviceptr_t` **srcDevice**
- `hipArray *` **srcArray**
- `size_t` **srcPitch**
- `size_t` **dstXInBytes**
- `size_t` **dstY**
- `hipMemoryType` **dstMemoryType**
- `void *` **dstHost**
- `hipDeviceptr_t` **dstDevice**
- `hipArray *` **dstArray**
- `size_t` **dstPitch**
- `size_t` **WidthInBytes**
- `size_t` **Height**

5.31 HIP_MEMCPY3D Struct Reference

Public Attributes

- `unsigned int` **srcXInBytes**
- `unsigned int` **srcY**
- `unsigned int` **srcZ**
- `unsigned int` **srcLOD**
- `hipMemoryType` **srcMemoryType**
- `const void *` **srcHost**
- `hipDeviceptr_t` **srcDevice**
- `hipArray_t` **srcArray**
- `unsigned int` **srcPitch**
- `unsigned int` **srcHeight**
- `unsigned int` **dstXInBytes**
- `unsigned int` **dstY**
- `unsigned int` **dstZ**
- `unsigned int` **dstLOD**
- `hipMemoryType` **dstMemoryType**
- `void *` **dstHost**
- `hipDeviceptr_t` **dstDevice**
- `hipArray_t` **dstArray**
- `unsigned int` **dstPitch**
- `unsigned int` **dstHeight**
- `unsigned int` **WidthInBytes**
- `unsigned int` **Height**
- `unsigned int` **Depth**

5.32 HIP_RESOURCE_DESC_st Struct Reference

Public Attributes

- HIPResourcetype [resType](#)
- - union {
 - struct {
 - [hipArray_t](#) [hArray](#)
 - array**
 - struct {
 - [hipMipmappedArray_t](#) [hMipmappedArray](#)
 - mipmap**
 - struct {
 - [hipDeviceptr_t](#) [devPtr](#)
 - [hipArray_Format](#) [format](#)
 - unsigned int [numChannels](#)
 - size_t [sizeInBytes](#)
 - linear**
 - struct {
 - [hipDeviceptr_t](#) [devPtr](#)
 - [hipArray_Format](#) [format](#)
 - unsigned int [numChannels](#)
 - size_t [width](#)
 - size_t [height](#)
 - size_t [pitchInBytes](#)
 - pitch2D**
 - struct {
 - int **reserved** [32]
 - reserved**
 - res**
- unsigned int [flags](#)

5.32.1 Member Data Documentation

5.32.1.1 devPtr

[hipDeviceptr_t](#) HIP_RESOURCE_DESC_st::devPtr
Device pointer

5.32.1.2 flags

unsigned int HIP_RESOURCE_DESC_st::flags
Flags (must be zero)

5.32.1.3 format

[hipArray_Format](#) HIP_RESOURCE_DESC_st::format
Array format

5.32.1.4 hArray

[hipArray_t](#) HIP_RESOURCE_DESC_st::hArray
HIP array

5.32.1.5 height

size_t HIP_RESOURCE_DESC_st::height
Height of the array in elements

5.32.1.6 hMipmappedArray

hipMipmappedArray_t HIP_RESOURCE_DESC_st::hMipmappedArray
HIP mipmapped array

5.32.1.7 numChannels

unsigned int HIP_RESOURCE_DESC_st::numChannels
Channels per array element

5.32.1.8 pitchInBytes

size_t HIP_RESOURCE_DESC_st::pitchInBytes
Pitch between two rows in bytes

5.32.1.9 resType

HIPresourcetype HIP_RESOURCE_DESC_st::resType
Resource type

5.32.1.10 sizeInBytes

size_t HIP_RESOURCE_DESC_st::sizeInBytes
Size in bytes

5.32.1.11 width

size_t HIP_RESOURCE_DESC_st::width
Width of the array in elements

5.33 HIP_RESOURCE_VIEW_DESC_st Struct Reference

Public Attributes

- HIPresourceViewFormat [format](#)
- size_t [width](#)
- size_t [height](#)
- size_t [depth](#)
- unsigned int [firstMipmapLevel](#)
- unsigned int [lastMipmapLevel](#)
- unsigned int [firstLayer](#)
- unsigned int [lastLayer](#)
- unsigned int **reserved** [16]

5.33.1 Detailed Description

Resource view descriptor

5.33.2 Member Data Documentation

5.33.2.1 depth

size_t HIP_RESOURCE_VIEW_DESC_st::depth
Depth of the resource view

5.33.2.2 firstLayer

unsigned int HIP_RESOURCE_VIEW_DESC_st::firstLayer
First layer index

5.33.2.3 firstMipmapLevel

unsigned int HIP_RESOURCE_VIEW_DESC_st::firstMipmapLevel
First defined mipmap level

5.33.2.4 format

HIPResourceViewFormat HIP_RESOURCE_VIEW_DESC_st::format
Resource view format

5.33.2.5 height

size_t HIP_RESOURCE_VIEW_DESC_st::height
Height of the resource view

5.33.2.6 lastLayer

unsigned int HIP_RESOURCE_VIEW_DESC_st::lastLayer
Last layer index

5.33.2.7 lastMipmapLevel

unsigned int HIP_RESOURCE_VIEW_DESC_st::lastMipmapLevel
Last defined mipmap level

5.33.2.8 width

size_t HIP_RESOURCE_VIEW_DESC_st::width
Width of the resource view

5.34 HIP_TEXTURE_DESC_st Struct Reference**Public Attributes**

- HIPAddress_mode [addressMode](#) [3]
- HIPfilter_mode [filterMode](#)
- unsigned int [flags](#)
- unsigned int [maxAnisotropy](#)
- HIPfilter_mode [mipmapFilterMode](#)
- float [mipmapLevelBias](#)
- float [minMipmapLevelClamp](#)
- float [maxMipmapLevelClamp](#)
- float [borderColor](#) [4]
- int [reserved](#) [12]

5.34.1 Detailed Description

Texture descriptor

5.34.2 Member Data Documentation

5.34.2.1 addressMode

HIPaddress_mode HIP_TEXTURE_DESC_st::addressMode
Address modes

5.34.2.2 borderColor

float HIP_TEXTURE_DESC_st::borderColor
Border Color

5.34.2.3 filterMode

HIPfilter_mode HIP_TEXTURE_DESC_st::filterMode
Filter mode

5.34.2.4 flags

unsigned int HIP_TEXTURE_DESC_st::flags
Flags

5.34.2.5 maxAnisotropy

unsigned int HIP_TEXTURE_DESC_st::maxAnisotropy
Maximum anisotropy ratio

5.34.2.6 maxMipmapLevelClamp

float HIP_TEXTURE_DESC_st::maxMipmapLevelClamp
Mipmap maximum level clamp

5.34.2.7 minMipmapLevelClamp

float HIP_TEXTURE_DESC_st::minMipmapLevelClamp
Mipmap minimum level clamp

5.34.2.8 mipmapFilterMode

HIPfilter_mode HIP_TEXTURE_DESC_st::mipmapFilterMode
Mipmap filter mode

5.34.2.9 mipmapLevelBias

float HIP_TEXTURE_DESC_st::mipmapLevelBias
Mipmap level bias

5.35 hipArray Struct Reference

Public Attributes

- void * **data**
- struct [hipChannelFormatDesc](#) **desc**
- unsigned int **type**
- unsigned int **width**
- unsigned int **height**
- unsigned int **depth**

- enum hipArray_Format **Format**
- unsigned int **NumChannels**
- bool **isDrv**
- unsigned int **textureType**

5.36 hipChannelFormatDesc Struct Reference

Public Attributes

- int **x**
- int **y**
- int **z**
- int **w**
- enum hipChannelFormatKind **f**

5.37 hipDeviceArch_t Struct Reference

Public Attributes

- unsigned [hasGlobalInt32Atomics](#): 1
32-bit integer atomics for global memory.
- unsigned [hasGlobalFloatAtomicExch](#): 1
32-bit float atomic exch for global memory.
- unsigned [hasSharedInt32Atomics](#): 1
32-bit integer atomics for shared memory.
- unsigned [hasSharedFloatAtomicExch](#): 1
32-bit float atomic exch for shared memory.
- unsigned [hasFloatAtomicAdd](#): 1
32-bit float atomic add in global and shared memory.
- unsigned [hasGlobalInt64Atomics](#): 1
64-bit integer atomics for global memory.
- unsigned [hasSharedInt64Atomics](#): 1
64-bit integer atomics for shared memory.
- unsigned [hasDoubles](#): 1
Double-precision floating point.
- unsigned [hasWarpVote](#): 1
Warp vote instructions (`__any`, `__all`).
- unsigned [hasWarpBallot](#): 1
Warp ballot instructions (`__ballot`).
- unsigned [hasWarpShuffle](#): 1
Warp shuffle operations. (`__shfl_`).*
- unsigned [hasFunnelShift](#): 1
Funnel two words into one with shift&mask caps.
- unsigned [hasThreadFenceSystem](#): 1
`__threadfence_system`.
- unsigned [hasSyncThreadsExt](#): 1
`__syncthreads_count`, `syncthreads_and`, `syncthreads_or`.
- unsigned [hasSurfaceFuncs](#): 1
Surface functions.
- unsigned [has3dGrid](#): 1
Grid and group dims are 3D (rather than 2D).
- unsigned [hasDynamicParallelism](#): 1
Dynamic parallelism.

5.38 hipDeviceProp_t Struct Reference

Public Attributes

- char [name](#) [256]
Device name.
- size_t [totalGlobalMem](#)
Size of global memory region (in bytes).
- size_t [sharedMemPerBlock](#)
Size of shared memory region (in bytes).
- int [regsPerBlock](#)
Registers per block.
- int [warpSize](#)
Warp size.
- int [maxThreadsPerBlock](#)
Max work items per work group or workgroup max size.
- int [maxThreadsDim](#) [3]
Max number of threads in each dimension (XYZ) of a block.
- int [maxGridSize](#) [3]
Max grid dimensions (XYZ).
- int [clockRate](#)
Max clock frequency of the multiProcessors in khz.
- int [memoryClockRate](#)
Max global memory clock frequency in khz.
- int [memoryBusWidth](#)
Global memory bus width in bits.
- size_t [totalConstMem](#)
Size of shared memory region (in bytes).
- int [major](#)
- int [minor](#)
- int [multiProcessorCount](#)
Number of multi-processors (compute units).
- int [l2CacheSize](#)
L2 cache size.
- int [maxThreadsPerMultiProcessor](#)
Maximum resident threads per multi-processor.
- int [computeMode](#)
Compute mode.
- int [clockInstructionRate](#)
- [hipDeviceArch_t](#) [arch](#)
Architectural feature flags. New for HIP.
- int [concurrentKernels](#)
Device can possibly execute multiple kernels concurrently.
- int [pciDomainID](#)
PCI Domain ID.
- int [pciBusID](#)
PCI Bus ID.
- int [pciDeviceID](#)
PCI Device ID.
- size_t [maxSharedMemoryPerMultiProcessor](#)
Maximum Shared Memory Per Multiprocessor.
- int [isMultiGpuBoard](#)

- 1 if device is on a multi-GPU board, 0 if not.
- int [canMapHostMemory](#)
 - Check whether HIP can map host memory.
- int [gcnArch](#)
 - DEPRECATED: use [gcnArchName](#) instead.
- char [gcnArchName](#) [256]
 - AMD GCN Arch Name.
- int [integrated](#)
 - APU vs dGPU.
- int [cooperativeLaunch](#)
 - HIP device supports cooperative launch.
- int [cooperativeMultiDeviceLaunch](#)
 - HIP device supports cooperative launch on multiple devices.
- int [maxTexture1DLinear](#)
 - Maximum size for 1D textures bound to linear memory.
- int [maxTexture1D](#)
 - Maximum number of elements in 1D images.
- int [maxTexture2D](#) [2]
 - Maximum dimensions (width, height) of 2D images, in image elements.
- int [maxTexture3D](#) [3]
 - Maximum dimensions (width, height, depth) of 3D images, in image elements.
- unsigned int * [hdpMemFlushCntl](#)
 - Addres of HDP_MEM_COHERENCY_FLUSH_CNTL register.
- unsigned int * [hdpRegFlushCntl](#)
 - Addres of HDP_REG_COHERENCY_FLUSH_CNTL register.
- size_t [memPitch](#)
 - Maximum pitch in bytes allowed by memory copies.
- size_t [textureAlignment](#)
 - Alignment requirement for textures.
- size_t [texturePitchAlignment](#)
 - Pitch alignment requirement for texture references bound to pitched memory.
- int [kernelExecTimeoutEnabled](#)
 - Run time limit for kernels executed on the device.
- int [ECCEnabled](#)
 - Device has ECC support enabled.
- int [tccDriver](#)
 - 1: If device is Tesla device using TCC driver, else 0
- int [cooperativeMultiDeviceUnmatchedFunc](#)
- int [cooperativeMultiDeviceUnmatchedGridDim](#)
- int [cooperativeMultiDeviceUnmatchedBlockDim](#)
- int [cooperativeMultiDeviceUnmatchedSharedMem](#)
- int [isLargeBar](#)
 - 1: if it is a large PCI bar device, else 0
- int [asicRevision](#)
 - Revision of the GPU in this device.
- int [managedMemory](#)
 - Device supports allocating managed memory on this system.
- int [directManagedMemAccessFromHost](#)
 - Host can directly access managed memory on the device without migration.
- int [concurrentManagedAccess](#)
 - Device can coherently access managed memory concurrently with the CPU.
- int [pageableMemoryAccess](#)
- int [pageableMemoryAccessUsesHostPageTables](#)
 - Device accesses pageable memory via the host's page tables.

5.38.1 Detailed Description

hipDeviceProp

5.38.2 Member Data Documentation

5.38.2.1 clockInstructionRate

```
int hipDeviceProp_t::clockInstructionRate
```

Frequency in khz of the timer used by the device-side "clock*" instructions. New for HIP.

5.38.2.2 cooperativeMultiDeviceUnmatchedBlockDim

```
int hipDeviceProp_t::cooperativeMultiDeviceUnmatchedBlockDim
```

HIP device supports cooperative launch on multiple devices with unmatched block dimensions

5.38.2.3 cooperativeMultiDeviceUnmatchedFunc

```
int hipDeviceProp_t::cooperativeMultiDeviceUnmatchedFunc
```

HIP device supports cooperative launch on multiple devices with unmatched functions

5.38.2.4 cooperativeMultiDeviceUnmatchedGridDim

```
int hipDeviceProp_t::cooperativeMultiDeviceUnmatchedGridDim
```

HIP device supports cooperative launch on multiple devices with unmatched grid dimensions

5.38.2.5 cooperativeMultiDeviceUnmatchedSharedMem

```
int hipDeviceProp_t::cooperativeMultiDeviceUnmatchedSharedMem
```

HIP device supports cooperative launch on multiple devices with unmatched shared memories

5.38.2.6 major

```
int hipDeviceProp_t::major
```

Major compute capability. On HCC, this is an approximation and features may differ from CUDA CC. See the arch feature flags for portable ways to query feature caps.

5.38.2.7 minor

```
int hipDeviceProp_t::minor
```

Minor compute capability. On HCC, this is an approximation and features may differ from CUDA CC. See the arch feature flags for portable ways to query feature caps.

5.38.2.8 pageableMemoryAccess

```
int hipDeviceProp_t::pageableMemoryAccess
```

Device supports coherently accessing pageable memory without calling hipHostRegister on it

5.39 hipExtent Struct Reference

Public Attributes

- `size_t width`
- `size_t height`
- `size_t depth`

5.40 hipExternalMemoryBufferDesc_st Struct Reference

Public Attributes

- unsigned long long **offset**
- unsigned long long **size**
- unsigned int **flags**

5.41 hipExternalMemoryHandleDesc_st Struct Reference

Public Attributes

- hipExternalMemoryHandleType **type**
- ```
union {
 int fd
 struct {
 void * handle
 const void * name
 } win32
} handle
```
- unsigned long long **size**
- unsigned int **flags**

## 5.42 hipFuncAttributes Struct Reference

### Public Attributes

- int **binaryVersion**
- int **cacheModeCA**
- size\_t **constSizeBytes**
- size\_t **localSizeBytes**
- int **maxDynamicSharedSizeBytes**
- int **maxThreadsPerBlock**
- int **numRegs**
- int **preferredShmemCarveout**
- int **ptxVersion**
- size\_t **sharedSizeBytes**

## 5.43 hipIpcEventHandle\_st Struct Reference

### Public Attributes

- char **reserved** [HIP\_IPC\_HANDLE\_SIZE]

## 5.44 hipIpcMemHandle\_st Struct Reference

### Public Attributes

- char **reserved** [HIP\_IPC\_HANDLE\_SIZE]

## 5.45 hipLaunchParams\_t Struct Reference

### Public Attributes

- void \* **func**  
*Device function symbol.*
- dim3 **gridDim**  
*Grid dimentions.*
- dim3 **blockDim**  
*Block dimentions.*
- void \*\* **args**  
*Arguments.*
- size\_t **sharedMem**  
*Shared memory.*
- hipStream\_t **stream**  
*Stream identifier.*

## 5.46 hipMemcpy3DParms Struct Reference

### Public Attributes

- hipArray\_t **srcArray**
- struct hipPos **srcPos**
- struct hipPitchedPtr **srcPtr**
- hipArray\_t **dstArray**
- struct hipPos **dstPos**
- struct hipPitchedPtr **dstPtr**
- struct hipExtent **extent**
- enum hipMemcpyKind **kind**

## 5.47 hipMipmappedArray Struct Reference

### Public Attributes

- void \* **data**
- struct hipChannelFormatDesc **desc**
- unsigned int **type**
- unsigned int **width**
- unsigned int **height**
- unsigned int **depth**
- unsigned int **min\_mipmap\_level**
- unsigned int **max\_mipmap\_level**
- unsigned int **flags**
- enum hipArray\_Format **format**

## 5.48 hipPitchedPtr Struct Reference

### Public Attributes

- void \* **ptr**
- size\_t **pitch**
- size\_t **xsize**
- size\_t **ysize**

## 5.49 hipPointerAttribute\_t Struct Reference

### Public Attributes

- enum hipMemoryType **memoryType**
- int **device**
- void \* **devicePointer**
- void \* **hostPointer**
- int **isManaged**
- unsigned **allocationFlags**

### 5.49.1 Detailed Description

Pointer attributes

## 5.50 hipPos Struct Reference

### Public Attributes

- size\_t **x**
- size\_t **y**
- size\_t **z**

## 5.51 hipResourceDesc Struct Reference

### Public Attributes

- enum hipResourceType **resType**
- ```

union {
    struct {
        hipArray\_t array
    } array
    struct {
        hipMipmappedArray\_t mipmap
    } mipmap
    struct {
        void * devPtr
        struct hipChannelFormatDesc desc
        size_t sizeInBytes
    } linear
    struct {
        void * devPtr
        struct hipChannelFormatDesc desc
        size_t width
        size_t height
        size_t pitchInBytes
    } pitch2D
} res

```

5.51.1 Detailed Description

HIP resource descriptor

5.52 hipResourceViewDesc Struct Reference

Public Attributes

- enum hipResourceViewFormat **format**
- size_t **width**
- size_t **height**
- size_t **depth**
- unsigned int **firstMipmapLevel**
- unsigned int **lastMipmapLevel**
- unsigned int **firstLayer**
- unsigned int **lastLayer**

5.52.1 Detailed Description

hip resource view descriptor

5.53 hipTextureDesc Struct Reference

Public Attributes

- enum hipTextureAddressMode **addressMode** [3]
- enum hipTextureFilterMode **filterMode**
- enum hipTextureReadMode **readMode**
- int **sRGB**
- float **borderColor** [4]
- int **normalizedCoords**
- unsigned int **maxAnisotropy**
- enum hipTextureFilterMode **mipmapFilterMode**
- float **mipmapLevelBias**
- float **minMipmapLevelClamp**
- float **maxMipmapLevelClamp**

5.53.1 Detailed Description

hip texture descriptor

5.54 int1 Union Reference

Public Attributes

- int **data**

5.55 int16 Union Reference

Public Attributes

- int **data** [16]

5.56 int2 Union Reference

Public Attributes

- int **data** [2]

5.57 int3 Union Reference

Public Attributes

- [int4](#) data

5.58 int4 Union Reference

Public Attributes

- int data [4]

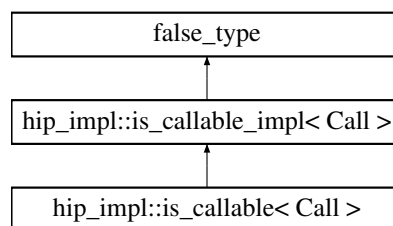
5.59 int8 Union Reference

Public Attributes

- int data [8]

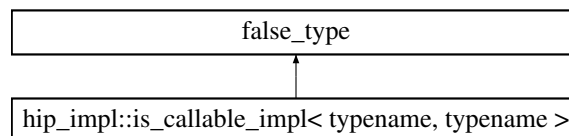
5.60 hip_impl::is_callable< Call > Struct Template Reference

Inheritance diagram for hip_impl::is_callable< Call >:



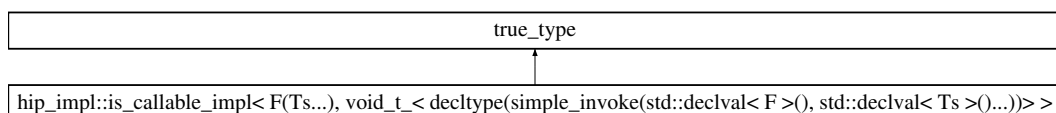
5.61 hip_impl::is_callable_impl< typename, typename > Struct Template Reference

Inheritance diagram for hip_impl::is_callable_impl< typename, typename >:



5.62 hip_impl::is_callable_impl< F(Ts...), void_t_< decltype(simple_invoke(std::declval< F >(), std::declval< Ts >()...))> > Struct Template Reference

Inheritance diagram for hip_impl::is_callable_impl< F(Ts...), void_t_< decltype(simple_invoke(std::declval< F >(), std::declval< Ts >()...))> >:



5.63 hip_impl::kernarg Class Reference

Public Member Functions

- **kernarg** ([kernarg](#) &&)
- `std::uint8_t * data ()`
- `std::size_t size ()`
- `void reserve (std::size_t)`
- `void resize (std::size_t)`
- **kernarg** ([kernarg](#) &&)
- `std::uint8_t * data ()`
- `std::size_t size ()`
- `void reserve (std::size_t)`
- `void resize (std::size_t)`

5.64 hip_impl::kernargs_size_align Class Reference

Public Member Functions

- `std::size_t size (std::size_t n) const`
- `std::size_t alignment (std::size_t n) const`
- `const void * getHandle () const`
- `std::size_t size (std::size_t n) const`
- `std::size_t alignment (std::size_t n) const`
- `const void * getHandle () const`

Friends

- [kernargs_size_align](#) `program_state::get_kernargs_size_align (std::uintptr_t)`
- [kernargs_size_align](#) `program_state::get_kernargs_size_align (std::uintptr_t)`

5.65 long1 Union Reference

Public Attributes

- `long data`

5.66 long16 Union Reference

Public Attributes

- `long data [16]`

5.67 long2 Union Reference

Public Attributes

- `long data [2]`

5.68 long3 Union Reference

Public Attributes

- [long4](#) `data`

5.69 long4 Union Reference

Public Attributes

- long **data** [4]

5.70 long8 Union Reference

Public Attributes

- long **data** [8]

5.71 longlong1 Union Reference

Public Attributes

- long long **data**

5.72 longlong16 Union Reference

Public Attributes

- long long **data** [16]

5.73 longlong2 Union Reference

Public Attributes

- long long **data** [2]

5.74 longlong3 Union Reference

Public Attributes

- [longlong4](#) **data**

5.75 longlong4 Union Reference

Public Attributes

- long long **data** [4]

5.76 longlong8 Union Reference

Public Attributes

- long long **data** [8]

5.77 hip_impl::program_state Class Reference

Public Member Functions

- **program_state** (const [program_state](#) &)=delete
- hipFunction_t **kernel_descriptor** (std::uintptr_t, hsa_agent_t)

- [kernargs_size_align](#) `get_kernargs_size_align` (std::uintptr_t)
- `hsa_executable_t load_executable` (const char *, const size_t, hsa_executable_t, hsa_agent_t)
- `hsa_executable_t load_executable_no_copy` (const char *, const size_t, hsa_executable_t, hsa_agent_t)
- `void * global_addr_by_name` (const char *name)
- `program_state` (const [program_state](#) &)=delete
- `hipFunction_t kernel_descriptor` (std::uintptr_t, hsa_agent_t)
- [kernargs_size_align](#) `get_kernargs_size_align` (std::uintptr_t)
- `hsa_executable_t load_executable` (const char *, const size_t, hsa_executable_t, hsa_agent_t)
- `hsa_executable_t load_executable_no_copy` (const char *, const size_t, hsa_executable_t, hsa_agent_t)
- `void * global_addr_by_name` (const char *name)

Friends

- class `agent_globals_impl`

5.78 short1 Union Reference

Public Attributes

- short `data`

5.79 short16 Union Reference

Public Attributes

- short `data` [16]

5.80 short2 Union Reference

Public Attributes

- short `data` [2]

5.81 short3 Union Reference

Public Attributes

- [short4](#) `data`

5.82 short4 Union Reference

Public Attributes

- short `data` [4]

5.83 short8 Union Reference

Public Attributes

- short `data` [8]

5.84 surfaceReference Struct Reference

Public Attributes

- hipSurfaceObject_t **surfaceObject**

5.84.1 Detailed Description

hip surface reference

5.85 TData Union Reference

Public Attributes

- __hip_float4_vector_value_type **f**
- __hip_int4_vector_value_type **i**
- __hip_uint4_vector_value_type **u**

5.86 textureReference Struct Reference

Public Attributes

- int **normalized**
- enum hipTextureReadMode **readMode**
- enum hipTextureFilterMode **filterMode**
- enum hipTextureAddressMode **addressMode** [3]
- struct [hipChannelFormatDesc](#) **channelDesc**
- int **sRGB**
- unsigned int **maxAnisotropy**
- enum hipTextureFilterMode **mipmapFilterMode**
- float **mipmapLevelBias**
- float **minMipmapLevelClamp**
- float **maxMipmapLevelClamp**
- hipTextureObject_t **textureObject**
- int **numChannels**
- enum hipArray_Format **format**

5.86.1 Detailed Description

hip texture reference

5.87 uchar1 Union Reference

Public Attributes

- unsigned char **data**

5.88 uchar16 Union Reference

Public Attributes

- unsigned char **data** [16]

5.89 uchar2 Union Reference

Public Attributes

- unsigned char **data** [2]

5.90 uchar2Holder Struct Reference

Public Attributes

- ```
union {
 unsigned int ui [2]
 unsigned char c [8]
};
```

## 5.91 uchar3 Union Reference

### Public Attributes

- [uchar4](#) **data**

## 5.92 uchar4 Union Reference

### Public Attributes

- unsigned char **data** [4]

## 5.93 uchar8 Union Reference

### Public Attributes

- unsigned char **data** [8]

## 5.94 ucharHolder Struct Reference

### Public Attributes

- ```
union {  
    unsigned char c [4]  
    unsigned int ui  
} __attribute__
```

5.95 uint1 Union Reference

Public Attributes

- unsigned int **data**

5.96 uint16 Union Reference

Public Attributes

- unsigned int **data** [16]

5.97 uint2 Union Reference

Public Attributes

- unsigned int **data** [2]

5.98 uint3 Union Reference

Public Attributes

- [uint4](#) **data**

5.99 uint4 Union Reference

Public Attributes

- unsigned int **data** [4]

5.100 uint8 Union Reference

Public Attributes

- unsigned int **data** [8]

5.101 ulong1 Union Reference

Public Attributes

- unsigned long **data**

5.102 ulong16 Union Reference

Public Attributes

- unsigned long **data** [16]

5.103 ulong2 Union Reference

Public Attributes

- unsigned long **data** [2]

5.104 ulong3 Union Reference

Public Attributes

- [ulong4](#) **data**

5.105 `ulong4` Union Reference

Public Attributes

- unsigned long **data** [4]

5.106 `ulong8` Union Reference

Public Attributes

- unsigned long **data** [8]

5.107 `ulonglong1` Union Reference

Public Attributes

- unsigned long long **data**

5.108 `ulonglong16` Union Reference

Public Attributes

- unsigned long long **data** [16]

5.109 `ulonglong2` Union Reference

Public Attributes

- unsigned long long **data** [2]

5.110 `ulonglong3` Union Reference

Public Attributes

- [ulonglong4](#) **data**

5.111 `ulonglong4` Union Reference

Public Attributes

- unsigned long long **data** [4]

5.112 `ulonglong8` Union Reference

Public Attributes

- unsigned long long **data** [8]

5.113 `ushort1` Union Reference

Public Attributes

- unsigned short **data**

5.114 ushort16 Union Reference

Public Attributes

- unsigned short **data** [16]

5.115 ushort2 Union Reference

Public Attributes

- unsigned short **data** [2]

5.116 ushort3 Union Reference

Public Attributes

- [ushort4](#) **data**

5.117 ushort4 Union Reference

Public Attributes

- unsigned short **data** [4]

5.118 ushort8 Union Reference

Public Attributes

- unsigned short **data** [8]

