

EAStdC 1.03.06 Index

namespace EA::StdC

v1.0

Luc Isaak
EAStdC written and maintained by:
Paul Pedriana

EAAignment.h	1
EABitTricks.h	1
EAByteCrackers.h	1
eactype.h	1
EADateTime.h	1
EAFixedPoint.h	1
EAGlobal.h	2
EAHashCRC.h	2
EAHashString.h	2
EAMathHelp.h	2
EAMemory.h	2
EARandom.h	2
EARandomDistribution.h	2
EAScanf.h	2
EASprintf.h	2
EAString.h	2
EATextUtil.h	2
Int128_t.h	2

EAAignment.h	
#define	EAAlignOf
size_t	AlignOf<T>
size_t	AlignOf<T>
T	AlignUp<T, size_t a>
T*	AlignUp<T, size_t a>
T	AlignUp<T>
T*	AlignUp<T>
T	AlignDown<T, size_t a>
T*	AlignDown<T, size_t a>
T	AlignDown<T>
T*	AlignDown<T>
size_t	GetAlignment<T>
bool	IsAligned<T, size_t a>
bool	IsAligned<T>
template<T, int count, int alignment>	
class	AlignedArray
template<T, int alignment>	
class	AlignedObject
uint16_t	ReadMisalignedUint16
uint32_t	ReadMisalignedUint32
uint64_t	ReadMisalignedUint64
void	WriteMisalignedUint16
void	WriteMisalignedUint32
void	WriteMisalignedUint64
EABitTricks.h	
// Bit manipulation	
T	TurnOffLowestBit<T>
T	IsolateLowestBit<T>
T	IsolateLowest0Bit<T>
T	GetTrailing0Bits<T>
T	GetTrailing1And0Bits<T>
T	PropagateLowestBitDownward<T>
T	TurnOffLowestContiguousBits<T>
T	TurnOnLowest0Bit<T>
unsigned	GetNextWithEqualBitCount
unsigned	IsolateSingleBits
unsigned	IsolateSingle0Bits
unsigned	IsolateSingle0And1Bits
int32_t	ShiftRightSigned
int	CountTrailing0Bits
int	CountLeading0Bits
int	CountBits
int	CountBits64
uint32_t	RotateLeft
uint32_t	RotateRight
uint32_t	ReverseBits
uint32_t	IsolateHighestBit
uint32_t	IsolateHighest0Bit
uint32_t	PropagateHighestBitDownward
uint32_t	GetHighestContiguous0Bits
T	GetBitwiseEquivalence<T>
bool	AreLessThan2BitsSet<T>
T	GetHighestBit<T>

bool	IsPowerOf2<T>
uint32_t	RoundUpToPowerOf2<T>
bool	IsPowerOf2Multiple<T,int n>
bool	IsPowerOf2Minus1<T>
bool	CrossesPowerOf2<T>
bool	CrossesPowerOf2<T, int n>
uint32_t	GetHighestBitPowerOf2
uint32_t	GetHighestBitPowerOf2
T	GetNextGreaterEven<T>
T	GetNextGreaterOdd<T>
T	RoundUpTo<T, int n>
int32_t	RoundUpToEx<T, int n>
T	RoundDownTo<T, int n>
T	RoundDownToEx<T, int n>
T	RoundUpToMultiple<T, int n>
T	RoundDownToMultiple<T, int n>
uint32_t	Log2
uint32_t	CeilLog2
// Overflow	
bool	SignedAdditionWouldOverflow<T>
bool	SignedSubtractionWouldOverflow<T>
bool	UnsignedAdditionWouldOverflow<T>
bool	UnsignedSubtractionWouldOverflow<T>
bool	UnsignedMultiplyWouldOverflow
bool	SignedMultiplyWouldOverflow
bool	UnsignedDivisionWouldOverflow
bool	SignedDivisionWouldOverflow
int	GetAverage
int	GetAverage_Ceiling
// Miscellaneous	
int	GetParity
bool	GetIsBigEndian
int	ToggleBetween0And1
T	ToggleBetweenIntegers<T>
bool	IsBetween0AndValue
void	ExchangeValues<T>
T	FloorMod<T>
int	GetSign
int	GetSignEx
int32_t	SignExtend12
int32_t	SignExtend24
bool	IsUnsigned<T>
#define	EAIIsUnsigned
bool	IsTwosComplement
bool	IsOnesComplement
bool	IsSignMagnitude
bool	IsOffsetBinary
#define	EAArarrayCount
#define	EAOffsetOf
EAByteCrackers.h	
Conventions:	
0 (zero) refers to the lowest byte,	
1 refers to the second lowest byte, etc.	
b means 8 bit byte	
w means 16 bit word	
d means 32 bit dword	
q means 64 bit quadword	

#define	UINT8_0_FROM_UINT16(w)
#define	UINT8_1_FROM_UINT16(w)
#define	UINT8_0_FROM_UINT32(d)
#define	UINT8_1_FROM_UINT32(d)
#define	UINT8_2_FROM_UINT32(d)
#define	UINT8_3_FROM_UINT32(d)
#define	UINT8_0_FROM_UINT64(q)
#define	UINT8_1_FROM_UINT64(q)
#define	UINT8_2_FROM_UINT64(q)
#define	UINT8_3_FROM_UINT64(q)
#define	UINT8_4_FROM_UINT64(q)
#define	UINT8_5_FROM_UINT64(q)
#define	UINT8_6_FROM_UINT64(q)
#define	UINT8_7_FROM_UINT64(q)
#define	UINT16_0_FROM_UINT32(d)
#define	UINT16_1_FROM_UINT32(d)
#define	UINT16_0_FROM_UINT64(q)
#define	UINT16_1_FROM_UINT64(q)
#define	UINT16_2_FROM_UINT64(q)
#define	UINT16_3_FROM_UINT64(q)
#define	UINT16_FROM_UINT8(b1, b0)
#define	UINT32_0_FROM_UINT64(q)
#define	UINT32_1_FROM_UINT64(q)
#define	UINT32_FROM_UINT8(b3, b2, b1, b0)
#define	UINT32_FROM_UINT16(w1, w0)
#define	UINT64_FROM_UINT8(b7, b6, b5, b4, b3, b2, b1, b0)
#define	UINT64_FROM_UINT16(w3, w2, w1, w0)
#define	UINT64_FROM_UINT32(d1, d0)
eactype.h	
int	Isalnum
int	Isalpha
int	Isdigit
int	Isxdigit
int	Isgraph
int	Islower
int	Isupper
int	Isprint
int	Isprint
int	Isspace
int	Iscntrl
int	Isascii
char_t	Tolower
char_t	Toupper(char_t c)
EADateTime.h	
class DateTime	
{	
DateTime(TimeFrame)	
DateTime(int64_t nSeconds)	

DateTime(uint32_t nYear, uint32_t nMonth, uint32_t nDayOfMonth, uint32_t nHour = 0, uint32_t nMinute = 0, uint32_t nSecond = 0)	
int	Compare(const DateTime&, bool bCompDate, bool bCompTime)
uint32_t	GetParameter
void	SetParameter
void	Set(TimeFrame)
void	Set(uint32_t nYear, uint32_t nMonth, uint32_t nDayOfMonth, uint32_t nHour, uint32_t nMinute, uint32_t nSecond)
void	AddTime
int64_t	GetSeconds
bool	IsLeapYear
uint32_t	GetDaysInYear
uint32_t	GetDaysInMonth
uint32_t	GetDayOfYear
int	Convert4DigitTo2DigitYear
int	Convert2DigitTo4DigitYear
uint32_t	GetCurrent
bool	IsDST
int64_t	GetDaylightSavingsBias
int64_t	GetTimeZoneBias
void	DateTimeToTm
void	TmToDateTime
void	DateTimeToFileTime
void	FileTimeToDateTime
void	DateTimeToSystemTime
void	SystemTimeToDateTime
bool	operator==
bool	operator>
bool	operator>=
bool	operator<
bool	operator<=
bool	operator!=
EAFixedPoint.h	
typedef int32_t	EAFixed16
// 16:16 fixed point (16 bits of fraction)	
typedef FTemplate<...>	SFixed16
typedef FTemplate<...>	UFixed16
#define	EAFixed16ToInt
#define	EAIIntToFixed16
#define	EAFixed16ToDouble
#define	EADoubleToFixed16
#define	EAFixed16ToFloat
#define	EAFloatToFixed16
#define	EAFixed16Negate
EAFixed16	EAFixed16Mul
EAFixed16	EAFixed16Div
EAFixed16	EAFixed16DivSafe
EAFixed16	EAFixed16MulDiv

EAFixed16	EAFixed16MulDivSafe
EAFixed16	EAFixed16Mod
EAFixed16	EAFixed16ModSafe
EAFixed16	EAFixed16Abs
template<T, int upShiftInt, int downShiftInt, int upMulInt, int downDivInt>	
struct FTemplate	
{	
void	FromFixed
T	AsFixed
int	AsInt
unsigned int	AsUnsignedInt
long	AsLong
unsigned long	AsUnsignedLong
float	AsFloat
double	AsDouble
bool	operator<
bool	operator>
bool	operator>=
bool	operator<=
bool	operator==
bool	operator!=
bool	operator~
FTemplate	operator+
FTemplate	operator-
FTemplate	operator*
FTemplate	operator/=
FTemplate	operator*=
FTemplate	operator/=
FTemplate	operator+=
FTemplate	operator-=
FTemplate	operator
FTemplate	operator&
FTemplate	operator^
FTemplate	operator^=
FTemplate	operator<<
FTemplate	operator>>
FTemplate	operator<<=
FTemplate	operator>>=
FTemplate	operator++
FTemplate	operator--
FTemplate	operator++
FTemplate	operator--
FTemplate	Abs
FTemplate	DivSafe
FTemplate	DivSafeAssign
static T	FixedMul
static T	FixedDiv
static T	FixedDivSafe
static T	FixedMulDiv
static T	FixedMulDivSafe
static T	FixedMod
static T	FixedModSafe
}	

