Low-Energy Accelerator (LEA) Common Parameter Blocks

Group 1 — basic pointwise vector/matrix operations

		input1	reserved1	vectorSize	input2	output	input1Offset	input2Offset	outputOffset	
		V_{S0}* =&**X**	occu.	Siz =N	V_{S1}* =&**Y**	V_D* =&**Z**	V_{S0}+=.. =X_i	V_{S1}+=.. =Y_i	V_D+=.. =Z_i	
LEACMD__ADDMATRIX	0x40	IQ16		u16	IQ16	IQ16	s16	s16	s16	$z_n = x_n + y_n$; n = 0...N-1
LEACMD__ADDLONGMATRIX	0x7C	IQ32		u16	IQ32	IQ32	s16	s16	s16	
LEACMD__MPYMATRIX	0x3C	Q.15		u16	Q.15	Q.15	s16	s16	s16	$z_n = x_n * y_n$; n = 0...N-1
LEACMD__MPYLONGMATRIX	0x74	Q.31		u16	Q.31	Q.31	s16	s16	s16	
LEACMD__MPYCOMPLEXMATRIX	0x78	cQ.15		u16	cQ.15	cQ.15	s16	s16	s16	
LEACMD__SUBMATRIX	0x70	IQ16		u16	IQ16	IQ16	s16	s16	s16	$z_n = x_n - y_n$; n = 0...N-1
LEACMD__SUBLONGMATRIX	0xD4	IQ32		u16	IQ32	IQ32	s16	s16	s16	

Group 2 — basic MAC vector

		input1	reserved1	vectorSize	input2	output	scaleFactor		
		V_{S0}* =&**X**	occu.	Siz =N	V_{S1}* =&**Y**	V_D* =&**Z**	SF		
LEACMD__MAC	0x54	Q.15		u16	Q.15	Q.31	rsvd.	$Z = \sum_{n=0}^{N-1}(x_n * y_n)$ n=0...N-1	on MAC3 $x_n \Leftrightarrow y_{n-1} \rightarrow z_0$ $x_n \Leftrightarrow y_n \rightarrow z_1$ $x_n \Leftrightarrow y_{n+1} \rightarrow z_2$
LEACMD__MAC3	0x134	Q.15		u16	Q.15	Q.31	rsvd.		
LEACMD__SCALEDMAC	0x138	Q.15		u16	Q.15	Q16.15	Q.15	...) * SF	on SCALEDMAC

Group 3 — MAC, point- FIR, correlation, convolution

		input1	reserved1	vectorSize	input2	output	input1Offset	input2Offset	outputOffset	
		V_{S0}* =&**X**	occu.	Siz =N	V_{S1}* =&**Y**	V_D* =&**Z**	V_{S0}+=.. =X_i	V_{S1}+=.. =Y_i	V_D+=.. =Z_i	
LEACMD__MACMATRIX	0x100	Q.15		u16	Q.15	Q.31	s16	s16	s16	$Z = \sum_{n=0}^{N-1}(x_n * y_n)$
LEACMD__MACLONGMATRIX	0xD0	Q.31		u16	Q.31	Q.31	s16	s16	rsvd.	
LEACMD__MACCOMPLEXMATRIX	0x68	cQ.15		u16	cQ.15	cQ.31	s16	s16	rsvd.	
LEACMD__MACCOMPLEXCONJUGATEMATRIX	0x6C	cQ.15		u16	cQ.15	cQ.31	s16	s16	rsvd.	

Group 4 — basic min/max vector search operations on 16B data

		input1	reserved1	vectorSize	output	
		V_{S0}* =&**X**	occu.	Siz =N	D* =&**Z**	
LEACMD__MAX	0x104	IQ16		u16	V,P	$z_0 = \max_n\{x_0,\dots,x_n\}$; $z_1 = n \xrightarrow[max]{} z_0$;
LEACMD__MAXUNSIGNED	0x10C	u16		u16	V,P	
LEACMD__MIN	0x108	IQ16		u16	V,P	$z_0 = \min_n\{x_0,\dots,x_n\}$; $z_1 = n \xrightarrow[min]{} z_0$;
LEACMD__MINUNSIGNED	0x110	u16		u16	V,P	

Group 5 — generic min/max search operations on 32B data

		input1	reserved1	vectorSize	output	input1Offset	reserved2	
		V_{S0}* =&**X**	occu.	Siz =N	D* =&**Z**	V_{S0}+=.. =X_i		
LEACMD__MAXLONGMATRIX	0xD8	IQ32		u16	V,P	s16	rsvd.	$z_0 = \max_n\{x_0,\dots,x_n\}$; $z_1 = n \xrightarrow[max]{} z_0$; n=0...N-1
LEACMD__MAXUNSIGNEDLONGMATRIX	0x114	u32		u16	V,P	s16	rsvd.	
LEACMD__MINLONGMATRIX	0xDC	IQ32		u16	V,P	s16	rsvd.	$z_0 = \min_n\{x_0,\dots,x_n\}$; $z_1 = n \xrightarrow[min]{} z_0$; n=0...N-1
LEACMD__MINUNSIGNEDLONGMATRIX	0x118	u32		u16	V,P	s16	rsvd.	

Group 6 — generic min/max search operations on dual 16B data /complex

		input1	reserved1	vectorSize	input1Offset	output	reserved2		
		V_{S0}* =&**X**	occu.	Siz =N	V_{S0}+=. =X_i	D* =&**Z**			
LEACMD__MAXMATRIX	0x44	IQ16		u16	s16	V_E,V_O,P_E,P_O	rsvd.	$z_0 = \max_n\{x_0,\dots,x_n\}$; $z_2 = n \xrightarrow[max]{} z_0$; n=0,2,4...N-2	$z_1 = \max_n\{x_1,\dots,x_n\}$; $z_3 = n \xrightarrow[max]{} z_1$; n=1,3,5...N-1
LEACMD__MAXUNSIGNEDMATRIX	0xE8	u16		u16	s16	V_E,V_O,P_E,P_O	rsvd.		
LEACMD__MINMATRIX	0x48	IQ16		u16	s16	V_E,V_O,P_E,P_O	rsvd.	$z_0 = \min_n\{x_0,\dots,x_n\}$; $z_2 = n \xrightarrow[min]{} z_0$; n=0,2,4...N-2	$z_1 = \min_n\{x_1,\dots,x_n\}$; $z_3 = n \xrightarrow[min]{} z_1$; n=1,3,5...N-1
LEACMD__MINUNSIGNEDMATRIX	0xEC	u16		u16	s16	V_E,V_O,P_E,P_O	rsvd.		

Group 7 — block- FIR, correlation, convolution

		input1	reserved1	vectorSize	coefs	output	filterLength	circularBufferMask	scaleFactor	
		V_{S0}* =&**X**	occu.	Siz =N	V_C* +&**H**	V_D* =&**Z**	Len =K	mask	SF	
LEACMD__FIR	0x18	Q.15		u16	Q.15	Q.15	u16	u16	rsvd.	$z_n = \sum_{k=0}^{K-1}(h_k * x_{K-k+n-1})$; n=0..N-1 (for mask=0xff)
LEACMD__FIRLONG	0xC8	Q.31		u16	Q.31	Q.31	u16	u16	rsvd.	
LEACMD__FIRCOMPLEX	0xE0	cQ.15		u16	cQ.15	cQ.15	u16	u16	rsvd.	
LEACMD__FIRCOMPLEXLONG	0x124	cQ.31		u16	cQ.31	cQ.31	u16	u16	rsvd.	
LEACMD__SCALEDFIR	0x13C	Q.15		u16	Q.15	Q.15	u16	u16	Q.15	...) * SF on SCALEDFIR

Group 8 — taylor functions on pointwise vectors/matrices

		input1	reserved1	vectorSize	output	coefs	order	scaleFactor		
		V_{S0}* =&**X**	occu.	Siz =N	V_D* =&**Z**	V_C* +&**C**	Ord =M	SF		
LEASCCMD__POLYNOMIAL	0x20	Q.15		u16	Q.15	Q.15	u16	IQ16.15		$z_n = SF\sum_{m=0}^{M-1}(C_m * x_n{}^m)$ n=0..N-1
LEASCCMD__POLYNOMIALLONG	0x30	Q.31		u16	Q.31	Q.31	u16	s16	rsvd.	

Group 9 — FFT, iFFT, bank filtering (DIT_Type)

		input1	reserved1	vectorSizeBy2	log2Size	
		V_{S0}* =&**X**	occu.	Siz =N/2	lb(N)	
LEACMD__FFTCOMPLEXAUTOSCALING	0xC0	cQ.15		u16	u16	$\boldsymbol{X} = FFT(x_0,\dots,x_{N-1})$ with... bit reverse sorted input
LEACMD__FFTCOMPLEXFIXEDSCALING	0x10	cQ.15		u16	u16	SF=> REG_{DST} on AUTOSCALING
LEACMD__FFTCOMPLEXLONG	0x9C	cQ.31		u16	u16	

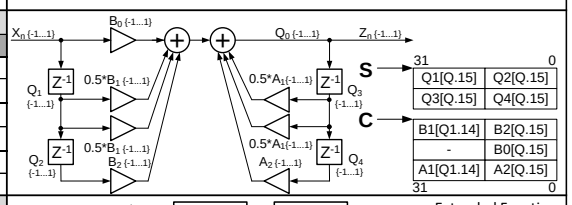
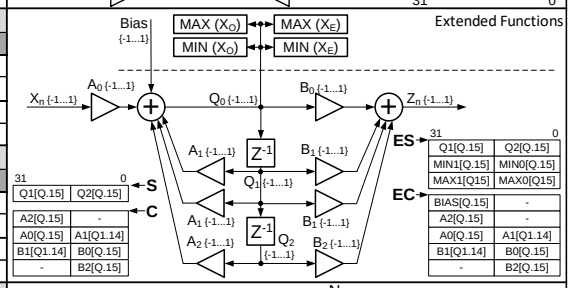
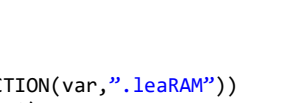
Group 10 — bit reversed carry propagated addressing pre-sort for DIT-FFTs

		input1	reserved1	sqryVectorSize	reserved2	
		V_{S0}* =&**X**	occu.	$\sqrt{N}$		
LEACMD__BITREVERSECOMPLEXEVEN	0x84	cQ.15		u16	rsvd.	$\boldsymbol{X} = bit-reverse(x_0,\dots,x_{N-1})$ n=0...N-1 with n+= $c \xrightarrow[Bit_{15}]{} \dots c \xrightarrow[Bit_{\sqrt{N}+1}]{} c \xrightarrow[Bit_{\sqrt{N}}]{} c \xrightarrow[Bit_{\sqrt{N}-1}]{} \dots c \xrightarrow[Bit_0]{}$
LEACMD__BITREVERSECOMPLEXODD	0x88	cQ.15		u16	rsvd.	
LEACMD__BITREVERSECOMPLEXLONGEVEN	0xAC	cQ.31		u16	rsvd.	
LEACMD__BITREVERSECOMPLEXLONGODD	0xB4	cQ.31		u16	rsvd.	

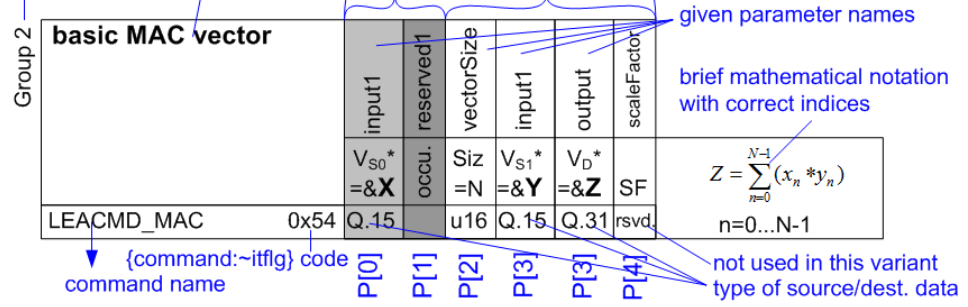
Group 12 — Vector/Matrix - Deinterleave/Sort Functions

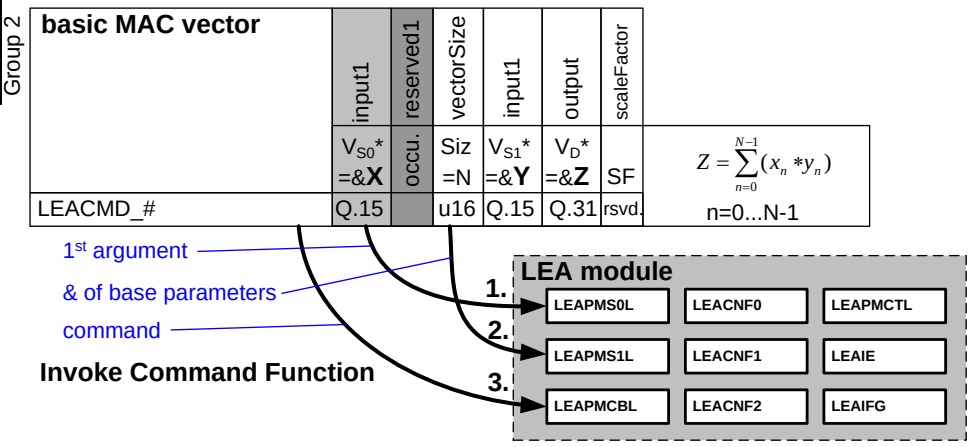
		input1	reserved1	vectorSize	interleaveDepth	output	reserved2	
		V_{S0}* =&**X**	occu.	Siz =N	=M	V_D* =&**Z**		
LEACMD__DEINTERLEAVEEVENEVEN	0x58	IQ16		u16	u16	IQ16	rsvd.	$z_n = x_{n*M}$ n=0..N-1
LEACMD__DEINTERLEAVEEVENODD	0x5C	IQ16		u16	u16	IQ16	rsvd.	
LEACMD__DEINTERLEAVEODDEVEN	0x60	IQ16		u16	u16	IQ16	rsvd.	
LEACMD__DEINTERLEAVEODDODD	0x64	IQ16		u16	u16	IQ16	rsvd.	
LEACMD__DEINTERLEAVELONG	0x12C	IQ32		u16	u16	IQ32	rsvd.	

Group B — Specials for Math, Matrix, DSP

LEACMD__MPYMATRIXROW **NxM**		input1; &N_{ROW} of Q.15	$[n_0 \ n_1 \ \dots \ n_{R-1}] \times \begin{bmatrix} m_{0,0} & m_{0,1} & m_{0,2} & m_{0,C-1} \\ m_{1,0} & m_{1,1} & m_{1,2} & m_{1,C-1} \\ m_{R-1,0} & m_{R-1,1} & m_{R-1,2} & m_{R-1,C-1} \end{bmatrix} = [z_0 \dots z_{C-1}]$ $N_{row\_of\_submatrix} \times M = Z_{row\_of\_submatrix}$
		reserved 1	
		rowSize; R as u16 (rows of M)	
		colSize; C u16 (cols of M)	
		colVector; &M_{COL} of Q.15	
	0x34	output &Z_{OUT} of Q.15	
LEACMD__POLYNOMIALSCALAR		input1; **X** as Q.31	$LEAPMDST = \sum_{m=0}^{M-1}(C_m * X^m) * SF;$
		coeffs ;&**C** of Q.31	
		order ;M	
	0xBC	scaleFactor ; SF as signed	
LEACMD__IIRBQ1 *Direct form I structure*		input1; &**X** of Q.15	
		reserved 1	
		vectorSize; N of u16	
		output; &**Z** of Q.15	
		state; &**S** of Q.15	
		coeffs; &**C** of Q.15	
		direction; S16	
	0x4C	reserved	
LEACMD__IIRBQ2 *Direct form II structure*		input1; &**X** of Q.15	
		reserved 1	
		vectorSize; N ofu16	
		output; &**Z** of Q.15	
		state; &S of Q.15	
	0x11C	coeffs; &C of Q.15	
LEACMD__IIRBQ2EXTENDED *Direct form II structure*		input1; &**X** of Q.15	
		reserved 1	
		vectorSize; N ofu16	
		output; &**Z** of Q.15	
		state; &ES of Q.15	
	0x90	coeffs; &EC of Q.15	
LEACMD__WINDOW		input1; &**X** of Q.15	*for n=0...K-1* $x_{N-n} = x_{N-n} * c_n$ and $x_n = x_n * c_n$ 
		reserved 1	
		vectorSize; N of u16	
		tapSize; K of u16	
	0x130	coeffs; &**C** of Q.15	

Group A





code snippet (actually complete test program)

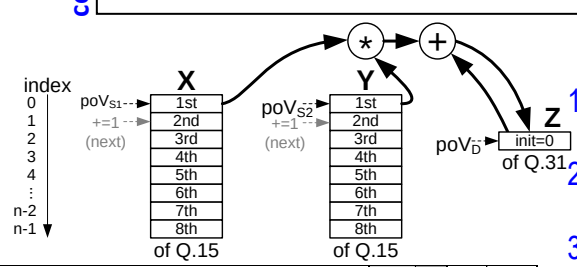
```
#include <msp430.h>

//macros to simplify LEA code (usually put in a header file)
#define LEA_EPRG(x) _Pragma(#x) //aux. def for NewLEA…
#define NewLEA_VAR(var) LEA_EPRG(RETAIN(var)) LEA_EPRG(DATA_SECTION(var,".leaRAM"))
#define Ladr(x) ((unsigned short)((unsigned long)(x) & 0xffff)>>2)
#define Q15(x) (x)<1.0? ((x)>-1.0? (x)*0x8000:0x8000):0x7FFF
#define LeaSP 0x3C00/4 //define Lea-Stack to top of LEA memory {32 bit oriented adr.}

NewLEA_VAR(Xi); NewLEA_VAR(Yi); NewLEA_VAR(Zo); NewLEA_VAR(P); //decl. variables in leaRAM

signed short Xi[]={Q15(-0.5), Q15(0.3), Q15(-0.4), Q15(0.5)}; //init with some input data
signed short Yi[]={Q15(-1)  , Q15(-1) , Q15(-1)  , Q15(-1) }; //init with some input data
signed long Zo[]={0x00000000};   //init space for result output
short P[]={0,0,4,0,0,0};         //here pre initialized parameters with N=4

void main(void) {
  WDTCTL = WDTPW | WDTHOLD;      // Stop watchdog timer
  LEACNF0=0; LEACNF1=0; LEACNF2=LeaSP;    //init LEA
  LEAPMCTL|=LEACMDEN;                     //enable LEA
  P[0]=Ladr(&Xi);  //set Xi in parameters
  P[3]=Ladr(&Yi);  //ditto with Yi using macro;
  P[4]=Ladr(&Zo);  //address of Zo to last element of parameters
  LEAPMS0L=P[0];   //put 1st parameter directly in register ~PMS0L; here adr of X[]
  LEAPMS1L=Ladr(&P[2]);  //put address of base parameter in ~PMS1L…
  LEAPMCBL=LEACMD__MAC + 0;  //start/invoke command with ~ITFLGS=0 for simple execution
  for(;;);}                  //have a look yourself
```



Invoke a command

1. Declare a parameter block and variables using *NewLEA_VAR* macros;
2. Initialize parameters according to the desired function (see this sheet)
3. Invoke command by writing to LEAPMS0L, LEAPMS1L and LEAPMCBL with proper mode by setting ~ITFLGS (see Invoke Methods)
4. Pick up results

LEA notifies completion using flags, events, and interrupts

Group 11 — FFT, post operation for real points

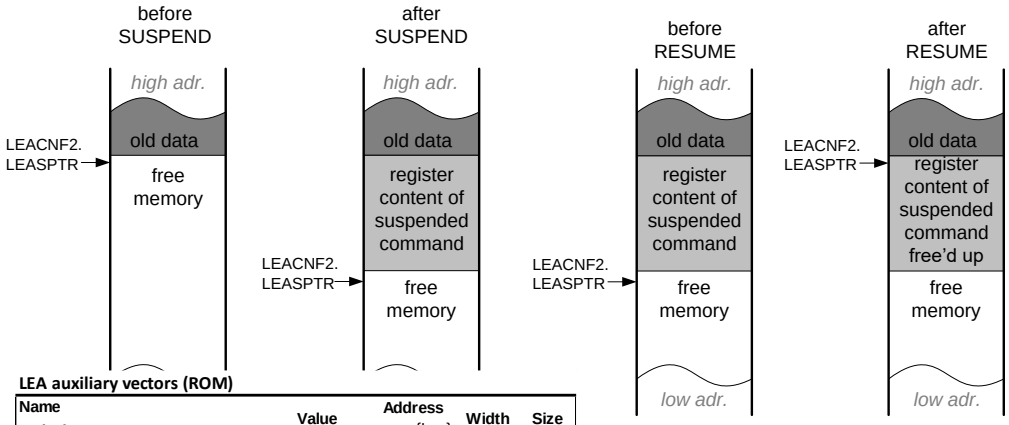
		input1	reserved1	vectorSize	log2Size
		V_{S0}* =&**X**	occu.	Siz =N	lb(N)
LEACMD__FFT (FFTSPLIT)	0x28	Q.15		u16	u16
LEACMD__FFTLONG (FFTLONGSPLIT)	0xA4	Q.31		u16	u16

Low-Energy Accelerator (LEA) Common Parameter Blocks

Group A: Programming Structure		input	vectorSize
		V_{S0}* =&**L**	N
LEACMD__MOVELONGLIST	0x80	u32	u16
LEACMD_SUSPEND	0x00		
LEACMD_RESUME	0x08		

MOVELONGLIST Parameter Structure

L → &SRC[0] {LeaAdr}, &DST[0] {LeaAdr}, &SRC[1] {LeaAdr}, &DST[1] {LeaAdr}, ⋮, &SRC[N-1] {LeaAdr}, &DST[N-1] {LeaAdr} (N*2)



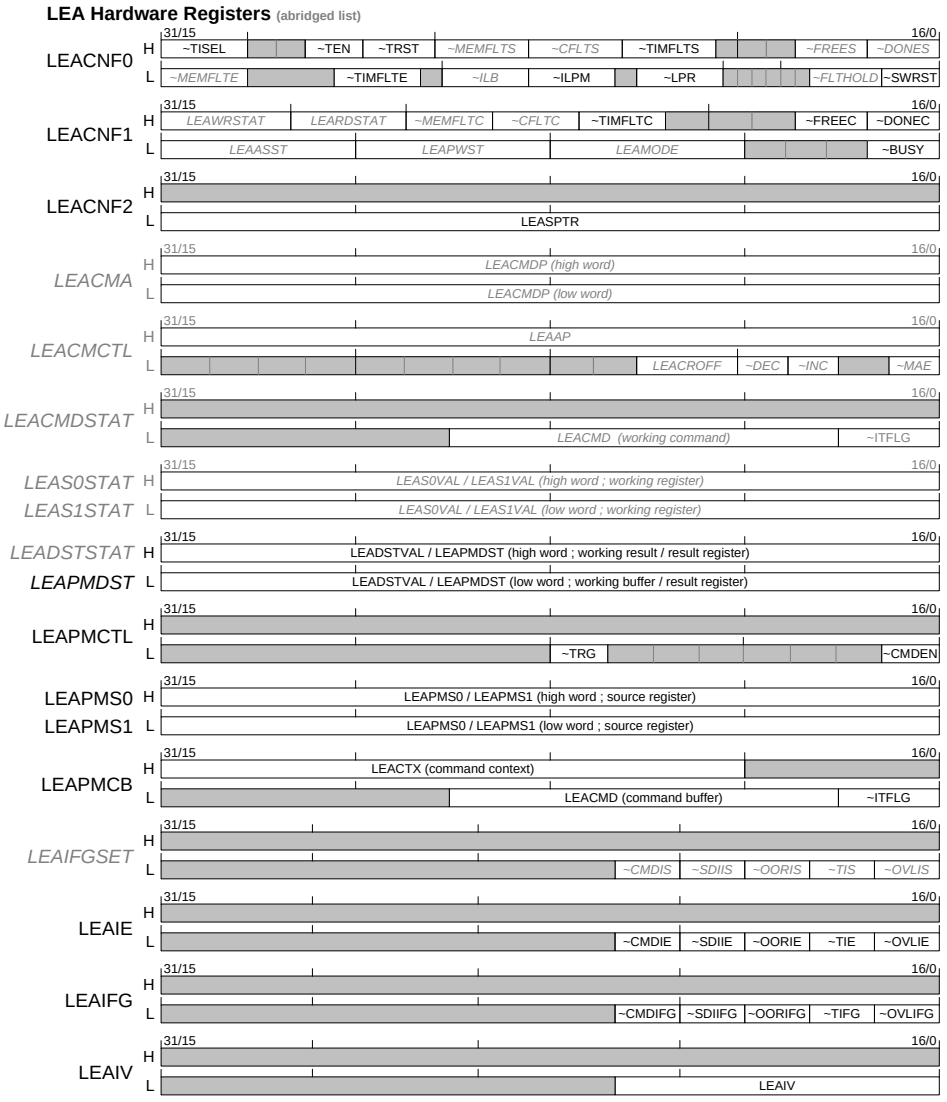
LEA auxiliary vectors (ROM)

Name *typical use*	Value	Address range{Lea}	Width	Size
LEA_CONST_0 *neutral element, dummy source*	0x0	0xA000.. 0xA3FF	16B (dual)	2048
LEA_CONST_1 *pos. epsilon*	0x1	0xA400.. 0xA7FF	16B (dual)	2048
LEA_CONST_4 *pos. amplitude margin 16B, + 0.5*	0x4000	0xA800.. 0xABFF	16B (dual)	2048
LEA_CONST_7 *neutral element, +1*	0x7FFF	0xAC00.. 0xAFFF	16B (dual)	2048
LEA_CONST_8 *substitutions, complementary element, conversion of ADC/DAC formats, -1,…*	0x8000	0xB000.. 0xB3FF	16B (dual)	2048
LEA_CONST_C *neg. amplitude margin 16B, -.5*	0xC000	0xB400.. 0xB7FF	16B (dual)	2048
LEA_CONST_E *neg. double epsilon, inv. element*	0xFFFE	0xB800.. 0xBBFF	16B (dual)	2048
LEA_CONST_F *neg. epsilon*	0xFFFF	0xBC00.. 0xBFFF	16B (dual)	2048
LEA_CONST_LONG_0 *neutral element, dummy source*	0x0	0xC000.. 0xC7FF	32B	2048
LEA_CONST_LONG_1 *pos. epsilon*	0x1	0xC800.. 0xCFFF	32B	2048
LEA_CONST_LONG_4 *pos. amplitude margin 32B, +.5*	0x4000:0000	0xD000.. 0xD7FF	32B	2048
LEA_CONST_LONG_7 *neutral element*	0x7FFF:FFFF	0xD800.. 0xDFFF	32B	2048
LEA_CONST_LONG_8 *substitutions, complementary element, conversion of ADC/DAC formats,…*	0x8000:0000	0xE000.. 0xE7FF	32B	2048
LEA_CONST_LONG_C *neg. amplitude margin 32B, -.5*	0xC000:0000	0xE800.. 0xEFFF	32B	2048
LEA_CONST_LONG_E *neg. double epsilon, inv. element*	0xFFFF:FFFE	0xF000.. 0xF7FF	32B	2048
LEA_CONST_LONG_F *neg. epsilon*	0xFFFF:FFFF	0xF800.. 0xFFFF	32B	2048

LEA Stack Frame Structure

Memory Address	Register Content Stored
LEASPTR + 100	LEADSTSTAT
LEASPTR + 96	LEACNF0
LEASPTR + 92	LEACNF1
LEASPTR + 88	LEACMDSTAT
LEASPTR + 84	LEAS1STAT
LEASPTR + 80	LEAS0STAT
LEASPTR + 76	PC-Saved
LEASPTR + 72	LoopEnd1 : LoopEnd0
LEASPTR + 68	LoopStart1 : LoopStart0
LEASPTR + 64	LoopCount1 : LoopCount0
LEASPTR + 60	Loop Index
LEASPTR + 56	Address Registers [SCL:SOV]
LEASPTR + 52	Address Registers [SR1:SR]
LEASPTR + 48	Address Registers [MR:PR]
LEASPTR + 44	Address Registers [PA3:PA2]
LEASPTR + 40	Address Registers [PA1:PA0]
LEASPTR + 36	Address Registers [MA:SA2]
LEASPTR + 32	Address Registers [SA1:SA0]
LEASPTR + 28	Data path register C7
LEASPTR + 24	Data path register C6
LEASPTR + 20	Data path register C5
LEASPTR + 16	Data path register C4
LEASPTR + 12	Data path register C3
LEASPTR + 8	Data path register C2
LEASPTR + 4	Data path register C1
LEASPTR + 0	Data path register C0

LEA Hardware Registers (abridged list)

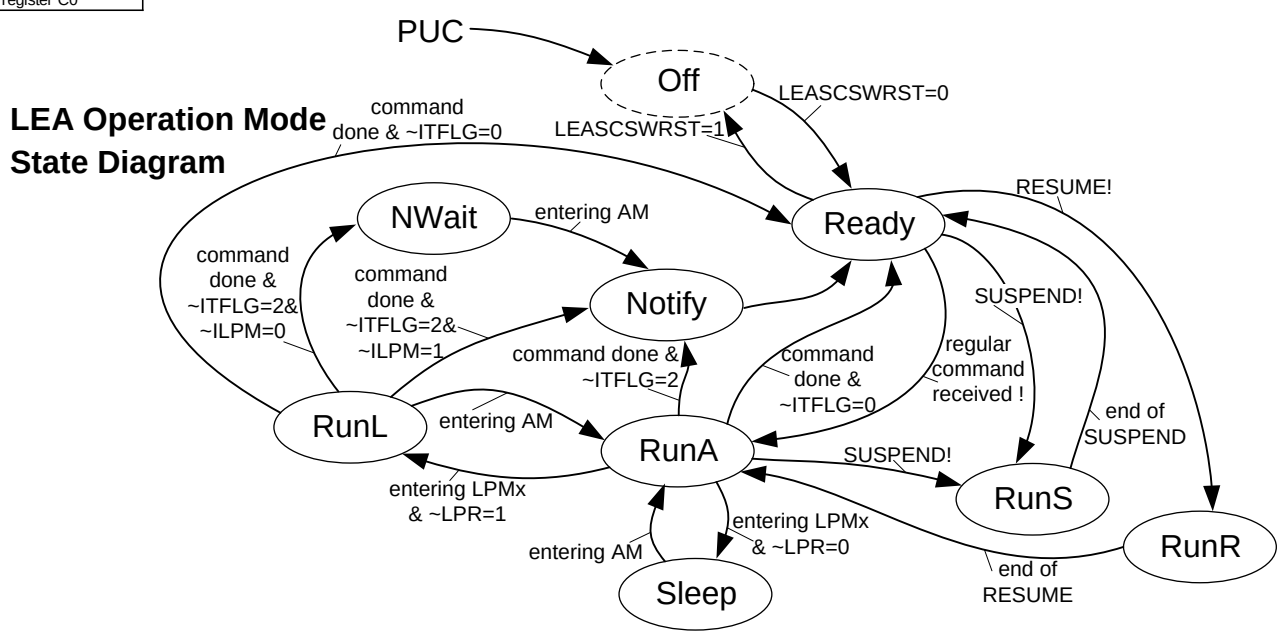


Registers mainly for Programming Purposes — *Registers mainly for Debug and Diagnostics Purposes*

LEA Command Invoke Methods

Invoke Method	LEA's Command Response
LEAPMCB=*LEACMD_xxx* ;	LEA command without any further indication
LEAPMCB= *LEACMD_xxx* +1;	LEA command with explicit result update
LEAPMCB= *LEACMD_xxx* +2;	LEA command with interrupt upon completion
LEAPMCB= *LEACMD_xxx* +3;	LEA command with interrupt and explicit result update

LEA Operation Mode State Diagram



LEA operation mode and behaviour

Operation Mode	accept CMD	PWR req	MCLK req	MODE
Notify (interims mode while setting interrupt)	no	yes	yes	0x5
Nwait (waiting for active mode)	n.a	yes	no	0x5
Off (optional mode only available on ULP architectures)	no	no	no	0x0
Ready (LEA waits for commands to execute)	yes	yes	no	0x1
RunA (execution mode of invoked command)	SUSPEND only	yes	yes	0x4
RunL (execution mode during low power mode)	n.a	yes	yes	0x7
RunR (LEACMD_RESUME execution)	no	yes	yes	0x3
RunS (LEACMD_SUSPEND execution)	no	yes	yes	0x2
Sleep (operations are stopped)	no	yes	no	0x6

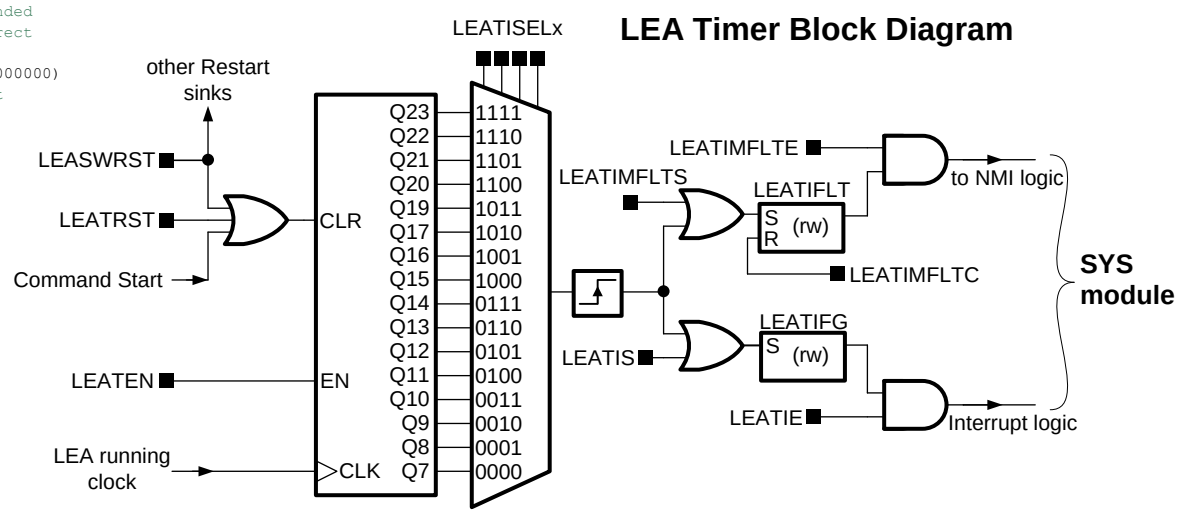
LEA comon functions polynomial coefficients (ROM)

Name	*function / purpose*	Address{Lea}	Width	Size N
LEA_CONST_LONG_SQRT	*coefficients for SQRT(X-1)*	0x8000	32B	33
LEA_CONST_LONG_EXP	*coefficients for EXP(X)*	0x8021	32B	13
LEA_CONST_LONG_2COS	*coefficients for COS(X^2)*	0x802E	32B	7
LEA_CONST_LONG_SIN	*coefficients for SIN(X)*	0x8035	32B	12
LEA_CONST_LONG_COS	*coefficients for COS(X)*	0x8041	32B	13
LEA_CONST_LONG_INV	*coefficients for 1/X*	0x804E	32B	33
LEA_CONST_LONG_2SECH	*coefficients for SECH(X^2)*	0x806F	32B	25
LEA_CONST_LONG_CBTF	*coefficients for CBRT(X) cubic root*	0x8088	32B	33
LEA_CONST_SQRTW	*coefficients for SQRT(X/PI) wide range*	0x80A1	16B	5
LEA_CONST_SQRT	*coefficients for SQRT(X-1)*	0x80A4	16B	13
LEA_CONST_SINW	*coefficients for SIN(X/PI)*	0x80AB	16B	7
LEA_CONST_SIN	*coefficients for SIN(X)*	0x80AF	16B	8
LEA_CONST_COS	*coefficients for COS(X)*	0x80B3	16B	9
LEA_CONST_INV	*coefficients for 1/X*	0x80B8	16B	17
LEA_CONST_TAN	*coefficient for TAN(X)*	0x80C1	32B	12
LEA_CONST_LONG_ASIN	*coefficient for ASIN(x)*	0x80CD	32B	24
LEA_CONST_LONG_ATN1	*coefficient for ATN(x) segment 1*	0x80E5	32B	6
LEA_CONST_LONG_ATN2	*coefficient for ATN(x) segment 2*	0x80EB	32B	9
LEA_CONST_LONG_ATN3	*coefficient for ATN(x) segment 3*	0x80F4	32B	10
LEA_CONST_LONG_2EXP	*coefficient for EXP(X^2)*	0x80FE	32B	10
LEA_CONST_LONG_LOG	*coefficient for LOG(X)*	0x8108	32B	13
LEA_CONST_LONG_4EXP	*coefficient for EXP(X) auxiliary*	0x8115	32B	10
LEA_CONST_LONG_2SIN	*coefficient for SIN(X^2)*	0x811F	32B	4

Useful macros for Code Composer

```
#define Ladr(x) ((unsigned short)((unsigned long)(x) & 0xffff)>>2)
#define Q15(x) (x)<1.0? ((x)>-1.0? (x)*0x8000:0x8000):0x7FFF            //Mathematical correct
#define Q15c(x) (((x)<1? ((x)>=-1 ? (x)*0x8000:0x8000):0x7FFF))         //Ceiling rounded version
#define Q15f(x) (x)>=0? ((x)>=1 ? 0x7FFF:(x)*0x7FFF):((x)<=-1? 0x8000:(x)*0x8000) //Floor rounded
#define Q31(x) (x)<1.0? ((x)>=-1.0 ? (x)*0x80000000:0x80000000):0x7FFFFFFF //Mathematical correct
#define Q31c(x) (((x)<1? ((x)>=-1 ? (x)*0x80000000:0x80000000):0x7FFFFFFF)) //Ceiling rounded
#define Q31f(x) (x)>=0? ((x)>=1.0? 0x7FFFFFFF:(x)*0x7FFFFFFF):((x)<=-1.0? 0x80000000:(x)*0x80000000)
#define IQ16_15(x) (x)<32768? ((x)>=-32768 ? (x)*0x8000:0x80000000):0x7FFFFFFF //Math. correct
#define LEA_EPRG(x) _Pragma(#x) // auxiliary macro
// constructor for new LEA global variable without alignment
#define NewLEA_VAR(var) LEA_EPRG(RETAIN(var)) LEA_EPRG(DATA_SECTION(var,".leaRAM"))
// constructor for new LEA variable with alignment
#define NewLEA_aVAR(var,algn) LEA_EPRG(RETAIN(var)) LEA_EPRG(DATA_SECTION(var,".leaRAM"))
LEA_EPRG(DATA_ALIGN(var,(algn)))
// constructor for new LEA scratch variable without alignment (API internal)
#define NewLEA_xVAR(var) LEA_EPRG(RETAIN(var)) LEA_EPRG(DATA_SECTION(var,".leaSCRATCH"))
// constructor for new global variable without alignment
#define NewLEA_dVAR(var) LEA_EPRG(RETAIN(var)) LEA_EPRG(DATA_SECTION(var,".data"))
// constructor for new FRAM variable without alignment
#define NewLEA fVAR(var) LEA EPRG(RETAIN(var)) LEA EPRG(DATA SECTION(var,".const"))
```

LEA Timer Block Diagram



IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale (https:www.ti.com/legal/termsofsale.html) or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265