

TML Instructions



Assignment

Syntax	Description	Code				Remarks
		0	1	2	3	
(V16D), TM = V16S	(V16D) from TM = V16S	0x90B0 TM	&V16D	&V16S		
(V16D), TM = V32S	(V16D) from TM = V32S	0x90B1 TM	&V16D	&V32S		
(V16D), TM = val16	(V16D) from TM = val16	0x90A0 TM	&V16D	val16		
(V16D), TM = val32	(V16D) from TM = val32	0x90A1 TM	&V16D	val32(L)	val32(H)	
(V16D+), TM = V16S	(V16D) from TM = V16S then V16D += 1	0x9030 TM	&V16D	&V16S		
(V16D+), TM = V32S	(V16D) from TM = V32S then V16D += 2	0x9031 TM	&V16D	&V32S		
(V16D+), TM = val16	(V16D) from TM = val16 then V16D += 1	0x9020 TM	&V16D	val16		
(V16D+), TM = val32	(V16D) from TM = val32 then V16D += 2	0x9021 TM	&V16D	val32(L)	val32(H)	
V16 = label	V16 = address of a TML label	0x2000 9LSB(&V16)	&label			Address inside page 0x0200
		0x2200 9LSB(&V16)	&label			Address inside page 0x0800
V16 = val16	V16 = val16	0x2000 9LSB(&V16)	val16			Address inside page 0x0200
		0x2200 9LSB(&V16)	val16			Address inside page 0x0800
V16D = (V16S), TM	V16D = (&V16S) from TM	0x9180 TM	&V16S	&V16D		
V16D = (V16S+), TM	V16D = (&V16S) from TM then V16S += 1	0x9100 TM	&V16S	&V16D		
V16D = V16S	V16D = V16S	0x2800 9LSB(&V16D)	&V16S			Address inside page 0x0200
		0x2A00 9LSB(&V16D)	&V16S			Address inside page 0x0800
V16D = -V16S	V16D = -V16S	0x3000 9LSB(&V16D)	&V16S			Address inside page 0x0200
		0x3200 9LSB(&V16D)	&V16S			Address inside page 0x0800
V16D = V32S(H)	V16D = V32S(H)	0x2800 9LSB(&V16D)	&V32S+1			Address inside page 0x0200
		0x2A00 9LSB(&V16D)	&V32S+1			Address inside page 0x0800
V16D = V32S(L)	V16D = V32S(L)	0x2800 9LSB(&V16D)	&V32S			Address inside page 0x0200
		0x2A00 9LSB(&V16D)	&V32S			Address inside page 0x0800
V16D, dm = V16S	V16D from dm = V16S (la)	0x9014	&V16D	&V16S		
V16D, dm = val16	V16 from dm = val16 (la)	0x9004	&V16D	val16		
V32 = val32	V32 = val32	0x2400 9LSB(&V32)	val32(L)	val32(H)		Address inside page 0x0200
		0x2600 9LSB(&V32)	val32(L)	val32(H)		Address inside page 0x0800
V32(H) = val16	V32(H) = val16	0x2000 9LSB(&V32+1)	val16			Address inside page 0x0200
		0x2200 9LSB(&V32+1)				Address inside page 0x0800
V32(L) = val16	V32(H) = val16	0x2000 9LSB(&V32)	val16			Address inside page 0x0200
		0x2200 9LSB(&V32)	val16			Address inside page 0x0800
V32D = (V16S), TM	V32D = (V16S) from TM	0x9181 TM	&V16S	&V32D		
V32D = (V16S+), TM	V32D = (V16S) from TM then V16D += 2	0x9101 TM	&V16S	&V32D		
V32D = V32S	V32D = V32S	0x2C00 9LSB(&V32D)	&V32S			V32D address inside page 0x0200
		0x2E00 9LSB(&V32D)	&V32S			V32D address inside page 0x0800
V32D = -V32S	V32D = -V32S	0x3400 9LSB(&V32D)	&V32S			V32D address inside page 0x0200
		0x3600 9LSB(&V32D)	&V32S			V32D address inside page 0x0800
V32D = V16S << N	V32D = V16S left-shifted by N	0x8960 N (N=0-16)	&V32D	&V16S		
V32D(H) = V16S	V32D(H) = V16	0x2800 9LSB(&V32D+1)	&V16S			V32D address inside page 0x0200
		0x2A00 9LSB(&V32D+1)	&V16S			V32D address inside page 0x0800
V32D(L) = V16S	V32D(L) = V16	0x2800 9LSB(&V32D)	&V16S			V32D address inside page 0x0200
		0x2A00 9LSB(&V32D)	&V16S			V32D address inside page 0x0800
V32D, dm = V32S	V32D from dm = V32S (la)	0x9015	&V32D	&V32S		
V32D, dm = val32	V32 from dm = val32 (la)	0x9005	&V32D	val32(L)	val32(H)	

Remarks:	
&V16 - 16 bits variable address	
&V32 - 32 bits variable address	
val16 - 16 bits value	
val32 - 32 bits value	
TM - type of memory: SPI (E2ROM) = 8	
	: DM (RAM Data Memory) = 4
	: PM (RAM Program Memory) = 0
Page 0x200: addresses from 0x200 to 0x3FF	
Page 0x800: addresses from 0x800 to 0x9FF	

Arithmetic&Logic

Syntax	Description	Code				Remarks
		0	1	2	3	
PROD <<= N	Left shift PROD by N (N=0-15)	0x88A0 N	0x030e			
PROD >>= N	Right shift PROD by N (N=0-15)	0x8880 N	0x030e			
SRB V16,ANDm,ORm	Set / Reset Bits of a V16	0x5800 9LSB(&V16)	ANDm	ORm		Address inside page 0x0200
		0x5A00 9LSB(&V16)	ANDm	ORm		Address inside page 0x0800
SRBL V16,ANDm,ORm	Set / Reset Bits of a V16 (long address)	0x5C00	&V16	ANDm	ORm	
V16 <<= N	Left shift V16 by N (N=0-15)	0x8820 N	&V16			
V16 * val16 << N	PROD = (V16 * val16) << N (N=0-15)	0x8C20 N	&V16	val16		
V16 * val16 >> N	PROD = (V16 * val16) >> N (N=0-15)	0x8C00 N	&V16	val16		
V16 += val16	Add val16 to V16	0x3800 9LSB(&V16)	val16			Address inside page 0x0200
		0x3A00 9LSB(&V16)	val16			Address inside page 0x0800
V16 -= val16	Subtract val16 from V16	0x4800 9LSB(&V16)	val16			Address inside page 0x0200
		0x4A00 9LSB(&V16)	val16			Address inside page 0x0800
V16 >>= N	Right shift V16 by N (N=0-15)	0x8800 N	&V16			
V16A * V16B << N	PROD = (V16A * V16B) << N (N=0-15)	0x8CA0 N	&V16A	&V16B		
V16A * V16B >> N	PROD = (V16A * V16B) >> N (N=0-15)	0x8C80 N	&V16A	&V16B		
V16D += V16S	Add V16S to V16D	0x4000 9LSB(&V16D)	&V16S			V16D address inside page 0x0200
		0x4200 9LSB(&V16D)	&V16S			V16D address inside page 0x0800
V16D -= V16S	Subtract V16S from V16D	0x5000 9LSB(&V16D)	&V16S			V16D address inside page 0x0200
		0x5200 9LSB(&V16D)	&V16S			V16D address inside page 0x0800
V32 * V16 << N	PROD = (V32 * V16) << N (N=0-15)	0x8DA0 N	&V32	&V16		
V32 * V16 >> N	PROD = (V32 * V16) >> N (N=0-15)	0x8D80 N	&V32	&V16		
V32 * val16 << N	PROD = (V32 * val16) << N (N=0-15)	0x8D20 N	&V32	val16		
V32 * val16 >> N	PROD = (V32 * val16) >> N (N=0-15)	0x8D00 N	&V32	val16		
V32 += val32	Add val32 to V32	0x3C00 9LSB(&V32)	val32(L)	val32(H)		Address inside page 0x0200
		0x3E00 9LSB(&V32)	val32(L)	val32(H)		Address inside page 0x0800
V32 <<= N	Left shift V32 by N (N=0-15)	0x8920 N	&V32			
V32 -= val32	Subtract val32 from V32	0x4C00 9LSB(&V32)	val32(L)	val32(H)		Address inside page 0x0200
		0x4E00 9LSB(&V32)	val32(L)	val32(H)		Address inside page 0x0800
V32 >>= N	Right shift V32 by N (N=0-15)	0x8900 N	&V32			
V32D += V32S	Add V32S to V32D	0x4400 9LSB(&V32D)	&V32S			V32D address inside page 0x0200
		0x4600 9LSB(&V32D)	&V32S			V32D address inside page 0x0800
V32D -= V32S	Subtract V32S from V32D	0x5400 9LSB(&V32D)	&V32S			V32D address inside page 0x0200
		0x5600 9LSB(&V32D)	&V32S			V32D address inside page 0x0800
V32 /=V16	Divide V32 by V16	0xDC01	&V32	&V16		

Remarks:	
&V16 - 16 bits variable address	
&V32 - 32 bits variable address	
val16 - 16 bits value	
val32 - 32 bits value	
Page 0x200: addresses from 0x200 to 0x3FF	
Page 0x800: addresses from 0x800 to 0x9FF	

Program Flow

Syntax	Description	Code				Remarks
		0	1	2	3	
ABORT	Abort function execution	0x1C02	&Label			
CALL V16	Unconditional CALL with address set in V16	0x7601	&V16			
CALL Label	Unconditional CALL of a function	0x7401	&Label			
CALL Label, V16, Flag	CALL function if V16 Flag 0	0x7401 Flag	&V16	&Label		
CALL Label, V32, Flag	CALL function if V32 Flag 0	0x7501 Flag	&V32	&Label		
CALLS V16	Cancelable CALL with address in set V16	0x1E01	&V16			
CALLS Label	Cancelable CALL of function	0x1C01	&Label			
GOTO V16	Unconditional GOTO with address set in V16	0x7600	&V16			
GOTO Label	Unconditional GOTO to label	0x7400	&Label			
GOTO Label, V16, Flag	GOTO label if V16 Flag 0	0x7400 Flag	&V16	&Label		
GOTO Label, V32, Flag	GOTO label if V32 Flag 0	0x7500 Flag	&V32	&Label		
RET	Return from TML function	0x0404				
RETI	Return from TML Interrupt SR	0x0504				

Flag	Value
LT	0x0090
LEQ	0x0088
EQ	0x00C0
NEQ	0x00A0
GT	0x0084
GEQ	0x0082

Remarks:	
&V16 - 16 bits variable address	
&V32 - 32 bits variable address	
&Label - label address	
Page 0x200: addresses from 0x200 to 0x3FF	
Page 0x800: addresses from 0x800 to 0x9FF	

Motion Mode

Syntax	Description	Code				Remarks
		0	1	2	3	
MODE CS	Set MODE Cam Slave	0x5909	0xB7C6	0x8706		
MODE GS	Set MODE Gear Slave	0x5909	0xB7C5	0x8705		
MODE PC	MODE Position Contouring	0x5909	0xBFC2	0x8702		
MODE PP	MODE Position Profile	0x5909	0xBFC1	0x8701		
MODE PSC	MODE S-Curve	0x5909	0xFFC1	0x8707		
MODE PT	MODE Position Time	0x5909	0xFFC1	0x870A		
MODE PVT	MODE Position Velocity Time	0x5909	0xFFC1	0x8709		
MODE SC	MODE Speed Contouring	0x5909	0xBBC2	0x8302		
MODE SE	MODE Speed External	0x5909	0xB3C0	0x8300		
MODE SP	MODE Speed Profile	0x5909	0xBBC1	0x8301		
MODE TC	MODE Torque Contouring	0x5909	0xB1C3	0x8103		
MODE TEF	MODE Torque External Fast loop	0x5909	0xB1E0	0x8120		
MODE TES	MODE Torque External Slow loop	0x5909	0xB1C0	0x8100		
MODE TT	MODE Torque Test	0x5909	0xB1C8	0x8108		
MODE VC	MODE Voltage Contouring	0x5909	0xB0C3	0x8003		
MODE VES	MODE Voltage External Slow loop	0x5909	0xB0C0	0x8000		
MODE VT	MODE Voltage Test	0x5909	0xB0C8	0x8008		

Event

Syntax	Description	Code				Remarks
		0	1	2	3	
IALPO V32	! if Absolute Load Position Over V32	0x7192	0x0228	&V32		
IALPO val32	! if Absolute Load Position Over val32	0x7092	0x0228	val32(L)	val32(H)	
IAMPO V32	! if Absolute Motor Position Over V32	0x7192	0x0988	&V32		
IAMPO val32	! if Absolute Motor Position Over val32	0x7092	0x0988	val32(L)	val32(H)	
IALPU V32	! if Absolute Load Position Under V32	0x7183	0x0228	&V32		
IALPU val32	! if Absolute Load Position Under val32	0x7083	0x0228	val32(L)	val32(H)	
IAMPU V32	! if Relative Position Under V32	0x7183	0x0988	&V32		
IAMPU val32	! if Relative Position Under val32	0x7083	0x0988	val32(L)	val32(H)	
IAT V32	! if Absolute Time >= V32	0x7198	0x02C0	&V32		
IAT val32	! if Absolute Time >= val32	0x7098	0x02C0	val32(L)	val32(H)	
ICAP	! if Capture triggered	0x700E				
IIN#n 0	! if Input #n is 0	0x70DB	Bit mask			Bit mask is set correspondingly with the IO used - if the mask has 1 on the bit number #n
IIN#n 1	! if Input #n is 1	0x70DA	Bit mask			
ILSN	! if Limit Switch Negative active	0x700C				
ILSP	! if Limit Switch Positive active	0x700D				
IMC	!(set event) if Motion Complete	0x700F				
IRO V32	! if Reference Over V32	0x7190	0x02AE	&V32		
IRO val32	! if Reference Over val32	0x7090	0x02AE	val32(L)	val32(H)	
IPRO V32	! if Position Reference Over V32	0x7190	0x02AE	&V32		
IPRO val32	! if Position Reference Over val32	0x7090	0x02AE	val32(L)	val32(H)	
ISRO V32	! if Speed Reference Over V32	0x7190	0x02AE	&V32		
ISRO val32	! if Speed Reference Over val32	0x7090	0x02AE	val32(L)	val32(H)	
ITRO V32	! if Torque Reference Over V32	0x7190	0x02AE	&V32		
ITRO val32	! if Torque Reference Over val32	0x7090	0x02AE	0x0000	val32(H)	Use only the high part of val32.
IRPO V32	! if Relative Position Over V32	0x7194	0x02BA	&V32		
IRPO val32	! if Relative Position Over val32	0x7094	0x02BA	val32(L)	val32(H)	
IRLPO V32	! if Relative Load Position Over V32	0x7194	0x02BA	&V32		
IRLPO val32	! if Relative Load Position Over val32	0x7094	0x02BA	val32(L)	val32(H)	
IRMPO V32	! if Relative Motor Position Over V32	0x7194	0x0988	&V32		
IRMPO val32	! if Relative Motor Position Over val32	0x7094	0x0988	val32(L)	val32(H)	
IRPU V32	! if Relative Position Under V32	0x7185	0x02BA	&V32		
IRPU val32	! if Relative Position Under val32	0x7085	0x02BA	val32(L)	val32(H)	
IRLPU V32	! if Relative Load Position Under V32	0x7185	0x02BA	&V32		
IRLPU val32	! if Relative Load Position Under val32	0x7085	0x02BA	val32(L)	val32(H)	
IRMPU V32	! if Relative Motor Position Under V32	0x7185	0x0988	&V32		
IRMPU val32	! if Relative Motor Position Under val32	0x7085	0x0988	val32(L)	val32(H)	
IRT V32	! if Relative Time >= V32	0x71B9	0x02C2	&V32		
IRT val32	! if Relative Time >= val32	0x70B9	0x02C2	val32(L)	val32(H)	
IRU V32	! if Reference Under V32	0x7181	0x02AE	&V32		
IRU val32	! if Reference Under val32	0x7081	0x02AE	val32(L)	val32(H)	
IPRU V32	! if Position Reference Under V32	0x7181	0x02AE	&V32		
IPRU val32	! if Position Reference Under val32	0x7081	0x02AE	val32(L)	val32(H)	
ISRU V32	! if Speed Reference Under V32	0x7181	0x02AE	&V32		
ISRU val32	! if Speed Reference Under val32	0x7081	0x02AE	val32(L)	val32(H)	
ITRU V32	! if Torque Reference Under V32	0x7181	0x02AE	&V32		
ITRU val32	! if Torque Reference Under val32	0x7081	0x02AE	0x0000	val32(H)	Use only the high part of val32.
ILSO V32	! if Load Speed Over V32	0x7196	0x098A	&V32		V32 is a variable of type fixed
ILSO val32	! if Load Speed Over val32	0x7096	0x098A	val32(L)	val32(H)	
IMSO V32	! if Motor Speed Over V32	0x7196	0x022C	&V32		V32 is a variable of type fixed
IMSO val32	! if Motor Speed Over val32	0x7096	0x022C	val32(L)	val32(H)	
ILSU V32	! if Load Speed Under V32	0x7187	0x098A	&V32		V32 is a variable of type fixed
ILSU val32	! if Load Speed Under val32	0x7087	0x098A	val32(L)	val32(H)	
IMSU V32	! if Motor Speed Under V32	0x7187	0x022C	&V32		V32 is a variable of type fixed
IMSU val32	! if Motor Speed Under val32	0x7087	0x022C	val32(L)	val32(H)	
IVO V32A, V32B	! if V32A Over V32B	0x7190	&V32A	&V32B		
IVO V32A, val32	! if V32A Over val32	0x7090	&V32A	val32(L)	val32(H)	
IVU V32A, V32B	! if V32A Under V32B	0x7181	&V32A	&V32B		
IVU V32A, val32	! if V32A Under val32	0x7081	&V32A	val32(L)	val32(H)	
WAIT!	Wait until event occurs	0x0408				
WAIT! Val32	Wait until event occurs, but exit if Time>val32	0x0608	val32(L)	val32(H)		

Remarks:	TML variably types
&V16 - 16 bits variable address	int - signed i 16 bits
&V32 - 32 bits variable address	long - signed 32 bits
val16 - 16 bits value	fixed - signed 16.16 bits
val32 - 32 bits value	
0x022C is the address of the variable fixed ASPD	
0x02AE is the address of the variable long TREF	
0x02BA is the address of the variable long RPOS	
0x02C2 is the address of the variable long RTIME	
0x0988 is the address of the variable long APOS_MT	
0x098A is the address of the variable fixed ASPD_LD	

Configuration and Command

Syntax	Description	Code					Remarks
		0	1	2	3	4	
AXISOFF	AXIS is OFF (deactivate control)	0x0002					
AXISON	AXIS is ON (activate control)	0x0102					
CPA	Command Position is Absolute	0x5909	0xFFFF	0x2000			
CPR	Command Position is Relative	0x5909	0xDFFF	0x0000			
DINT	Disable TML Interrupts	0x0410					
EINT	Enable TML Interrupts	0x0510					
ENDINIT	END of Initialization	0x0020					
EXTREF 0	External Reference read from ONLINE EREF	0x5909	0xFF3F	0x0000			
EXTREF 1	External Reference read from ANALOGUE	0x5909	0xFF7F	0x0040			
EXTREF 2	External Reference read from DIGITAL	0x5909	0xFFBF	0x0080			Enable 2nd encoder read
PTP P, T, C	Send a PT point &V32 (position), &T, C	0xC900 C	&P	&T			P-32bit variable, T-16bit variable
PTP val32, val16, C	Send a PT point val32, val16, C	0xC800 C	val32(L)	val32(H)	val16		C - 7 bit counter value
PVTP P, S, T, C	Send a PVT point &P, &S, &T, C	0x1800 C	&P	&S	&T		S -32bit variable
PVTP valP, valS, valT, C	Send a PVT point valP, valS, valT, C	0x1800 C	16valP	8valP 8valS	16valS	9valT	
REG_OFF	Registration mode disabled	0x5909	0xEFFF	0x0000			
REG_ON	Registration mode enabled	0x5909	0xFFFF	0x1000			
RESET	RESET DSP controller	0x0402					
RGM	Reset axis as Gear/Cam Master	0x5909	0xF7FF	0x0000			
SAP V32	Set Actual Position = V32	0x8000 9LSB(&V32)					Address inside page 0x0200
		0x8200 9LSB(&V32)					Address inside page 0x0800
SAP val32	Set Actual Position = val32	0x8400	val32(L)	val32(H)			
SEG D_time, D_ref	Segment D_time, D_ref	0x7800	D_time	D_ref(L)	D_ref(H)		
SEG V16, V32	Segment V16, V32	0x7C00 9LSB(&V16)	&V32				V16 Address inside page 0x0200
		0x7E00 9LSB(&V16)	&V32				V16 Address inside page 0x0800
SETPT val16	Configure PT mode	0x1880	val16				
SETPVT val16	Configure PVT mode	0x1880	val16				
SETSsync val32	Synchronization start/stop	0x1404	val32(L)	var32(H)			
SGM	Set axis as Gear/Cam Master	0x5909	0xFFFF	0x8800			
STA	Set Target pos. = Actual pos.	0x2CB2	0x0228				
STOP	STOP motion	0x01C4					
STOP!	STOP motion when ! (event occurs)	0x00C4					
TUM0	Set Target Update Mode 0	0x5909	0xBFFF	0x0000			
TUM1	Set Target Update Mode 1	0x5909	0xFFFF	0x4000			
UPD	Update motion immediate	0x0108					
UPD!	Update when ! (event occurs)	0x0008					

Remarks:
&V16 - 16 bits variable address
&V32 - 32 bits variable address
val16 - 16 bits value
val32 - 32 bits value
Page 0x200: addresses from 0x200 to 0x3FF
Page 0x800: addresses from 0x800 to 0x9FF

INputs & OUTputs

Syntax	Description	Code				Remarks
		0	1	2	3	
DIS2CAPI	Disable 2nd CAPI capture	0x04A0				
DISCAPI	Disable CAPI capture	0x0481				
DISLSN	Disable LSN limit switch	0x0681				
DISLSP	Disable LSP limit switch	0x06A0				
EN2CAPI0	Enable 2nd CAPI capture for 1->0	0x0420				
EN2CAPI1	Enable 2nd CAPI capture for 0->1	0x0520				
ENCAPI0	Enable CAPI capture for 1->0	0x0401				
ENCAPI1	Enable CAPI capture for 0->1	0x0501				
ENLSN0	Enable LSN limit switch for 1->0	0x0601				
ENLSN1	Enable LSN limit switch for 0->1	0x0701				
ENLSP0	Enable LSP limit switch for 1->0	0x0620				
ENLSP1	Enable LSP limit switch for 0->1	0x0720				
OUT(n,m,p) =V16	Set output#n, #m and #p with V16 value	0xED00	Bit mask	&V16		<i>Bit mask is set correspondigly with the IO used - ithe mask has 1 on bit number #n, #m and #p</i>
OUT(n,m,p) =val16	Set output#n, #m and #p with val16	0xEC00	Bit mask	val16		
V16D = IN(n,m,p)	Read inputs #n, #m and #p into V16D	0xE800	Bit mask	&V16D		
SetAsInput(n,m,p)	Set IO #n, #m and #p as inputs	0xEE00	Bit mask			
SetAsOutput(n,m,p)	Set IO#n, #m, and #p as output	0xEF00	Bit mask			

Remarks:

&V16 - 16 bits variable adress

val16 - 16 bits value

Multi-Axis

Syntax	Description	Code				Remarks
		0	1	2	3	
[A/G] { Instr1; }	Send TML instruction to [A/G]	0x9400 LengthMLI	A/G	MLI code	1-4 words	
[A/G] (V16D),TM = V16S	[A/G] (V16D),TM = local V16S	0x98B0 TM	A/G	& V16D	& V16S	
[A/G] (V16D),TM = V32S	[A/G] (V16D),TM = local V32S	0x98B1 TM	A/G	& V16D	& V32S	
[A/G] (V16D+),TM = V16S	[A/G] (V16D),TM = local V16S then V16D +	0x9830 TM	A/G	& V16D	& V16S	
[A/G] (V16D+),TM = V32S	[A/G] (V16D),TM = local V32S then V16D +	0x9831 TM	A/G	& V16D	& V32S	
[A/G] V16D = V16S	[A/G] V16D = local V16S	0xB800 9LSB(&V16D)	A/G	& V16S		Address inside page 0x0200
		0xBA00 9LSB(&V16D)	A/G	& V16S		Address inside page 0x0800
[A/G] V16D,dm = V16S	[A/G] V16D,dm = local V16S (la)	0x9814	A/G	& V16D	& V16S	
[A/G] V32D = V32S	[A/G] V32D = local V32S	0xBC00 9LSB(&V32D)	A/G	& V32S		Address inside page 0x0200
		0xBE00 9LSB(&V32D)				Address inside page 0x0800
[A/G] V32D,dm = V32S	[A/G] V32D,dm= local V32S (la)	0x9815	A/G	& V32D	& V32S	
ADDGRID V16	Add Group ID = V16	0x0940	&V16			
ADDGRID val16	Add Group ID = val16	0x0840	val16			There are maximum 8 groups
AXISID val16	AXIS ID = val16	0x0801	val16			There are maximum 255 axes
AXISID V16	AXIS ID = V16	0x0901	&V16			
REMGRID V16	Remove Group ID = V16	0x0980	&V16			
REMGRID val16	Remove Group ID = val16	0x0880	val16			There are maximum 8 groups
V16D = [A] (V16S),TM	Local V16D = [A] (V16S), dm	0x9D80 TM	A/G	& V16S	& V16D	
V16D = [A] (V16S+),TM	Local V16D = [A] (V16S), dm then V16S +=	0x9D00 TM	A/G	& V16S	& V16D	
V16D = [A] V16S	Local V16D = [A] V16S	0xE000 9LSB(&V16S)	A/G	& V16D		
		0xE200 9LSB(&V16S)				
V16D = [A] V16S,dm	Local V16D = [A] V16S, dm (la)	0x9C04	A/G	& V16S	& V16D	
V32D = [A] V32S,dm	Local V32D = [A] V32S, dm (la)	0x9C05	A/G	& V32S	& V32D	
V32D = [A] (V16S),TM	Local V32D = [A] (V16S),TM	0x9D81 TM	A/G	& V16S	& V32D	
V32D = [A] (V16S+),TM	Local V32D = [A] (V16S),TM then V16S +=	0x9D01 TM	A/G	& V16S	& V32D	
V32D = [A] V32S	Local V32D = [A] V32S	0xE400 9LSB(&V32S)	A/G	& V32D		Address inside page 0x0200
		0xE600 9LSB(&V32S)	A/G	& V32D		Address inside page 0x0800

Remarks:	
A/G in Code 1	: bit0 - bit3 = 0 (for MLI code)
	: bit4 - bit11 = A/G
	: bit12 = 0 (axis)
	= 1 (group)
	: bit13 - bit15 = reserved
&V16	- 16 bits variable address
&V32	- 32 bits variable address
val16	- 16 bits value
TM	- type of memory: SPI (E2ROM) = 8
	: DM (RAM Data Memory) = 4
	: PM (RAM Program Memory) = 0
Page 0x200:	addresses from 0x200 to 0x3FF
Page 0x800:	addresses from 0x800 to 0x9FF

Miscellaneous

Command sent to Drive/Intelligent Motor						
Syntax	Description	Code				Remarks
		0	1	2	3	
?V16	Give me data from 16 bit variable, from DM,	0xB000 TM	Expeditoer AxisID (master)	&V16		<i>The online instructions are intended only for host/master usage and cannot reside in a TML program. Therefore their syntax is fictive, the only goal being to identify these commands.</i>
?V32	Give me data from 32 bit variable, from DM, from one axis. On-line cmd.	0xB001 TM	Expeditoer AxisID (master)	&V32		
??V16	Give me data from 16 bit variable, from DM, from a group. On-line cmd.	0xB200 TM	Expeditoer AxisID (master)	&V16		
??V32	Give me data from 32 bit variable, from DM, from a group. On-line cmd.	0xB201 TM	Expeditoer AxisID (master)	&V32		
?TMLV16	Give me data from 16 bit TML variable. On-line cmd.	0xA000 9LSB(&V16TML)	Expeditoer AxisID (master)			
		0xA200 9LSB(&V16TML)				
?TMLV32	Give me data from 32 bit TML variable. On-line cmd.	0xA400 9LSB(&V32TML)	Expeditoer AxisID (master)			
		0xA600 9LSB(&V32TML)				
GETVER	Get version. On-line cmd.	0xD801	Expeditoer AxisID (master)			
PING val16	Ask a group of axes to return their axis ID	0xD600	ExpeditoerID H	val16		
PONG	Response to a PING	0xD600 AxisID	FwCode1 FwCode2	FwCode3 FwCode4		
BEGIN	BEGIN of a TML program	0x649C				
CHECKSUM, TM Start, Stop, V16D	addresses from TM	0xDB10 (TM<<3)	&V16D	Start	Stop	
END	END of a TML program	0x0001				
ENEEPROM	Enable EEPROM	0x9500				
FAULTR	Reset drive fault state	0x1C04				
INITCAM addrS, addrD	Copy CAM table from SPI (addrS address) to RAM (addrD address)	0xD8C0	addrS	addrD		
LOCKEEPROM var16	Write protect/unprotect EEPROM	0x9600 var16				var16= 0 - no protection, 1 - protect last quarter, 2 - protect last half, 3 - protect entire EEPROM
NOP	No Operation	0x0000				
SAVE	Save setup table in E2ROM	0x1C08				
SCIBR V16	Set SCI Baud Rate	0x0920	&V16			
SCIBR val16	Set SCI Baud Rate	0x0820	val16			
SPIBR V16	Set SPI Baud Rate	0x0910	&V16			
SPIBR val16	Set SPI Baud Rate	0x0810	val16			
CANBR val16	Set CAN-bus Baud-Rate	0x0804	val16			

Drive/Intelligent motor answer					
Syntax	Code				
	0	1	2	3	4
?V16	0xB400 TM	Expeditoer AxisID (slave)	&V16	16-bit data	
?V32	0xB401 TM	Expeditoer AxisID (slave)	&V32	16LSB of data	16MSB of data
??V16	0xD400 ExpeditoerID	&V16	16-bit data		
??V32	0xD500 ExpeditoerID	&V32	16LSB of data	16MSB of data	
?TMLV16	0xA800 9LSB(&V16TML)	Expeditoer AxisID (slave)	16-bit data		
?TMLV32	0xAC00 9LSB(&V32TML)	Expeditoer AxisID (slave)	16LSB of data	16MSB of data	
GETVER	0xD801	FWCODE	FWCODE		

Remarks:
&V16 - 16 bits variable address
&V32 - 32 bits variable address
val16 - 16 bits value
val32 - 32 bits value
H - Host bit (set 1 when sent via RS-232)
TM - type of memory:
SPI (E2ROM) = 8
DM (RAM Data Memory) = 4
PM (RAM Program Memory) = 0
FwCode1-4 - 4 bytes representing 4 ascii codes for firmware version