

TML interrupt routines usage

MOTION TECHNOLOGY

Application Note

Easy Motion Studio II



Your

Intelligent Move



Table of contents

1. Application description	. 2
2. Application flow chart	. 2
3. EasyMotion Studio II implementation	3
3.1 Time period interrupt routine	3
3.2 Main motion section	. 7
4. Application evaluation	. 9

1. Application description

This application note describes how to activate and customize the TML interrupt services routines, using an example that sets the "Int 10 - Time period has elapsed" interrupt, to flash a LED, connected to one of the drive digital outputs.

The TML interrupts are special conditions that are continuously monitored by the drive firmware. When a TML interrupt occurs, the main TML program execution is suspended and the TML code associated with the interrupt, called Interrupt Service Routine (in short ISR), is executed. While an interrupt is active (meaning the ISR code is being executed), all the TML interrupts are globally disabled.

That is why, it is recommended to put in the ISR (Interrupt Status Register) only the minimum needed code. If this is not possible, then other interrupts can be re-enabled using the "Interrupts Settings" dialogue (will be presented in chapter 3).

2. Application flow chart



3. EasyMotion Studio II implementation

The application implementation in EasyMotion Studio II, includes 2 main parts:

- 1) The interrupt ("int10") that contains the TML code to be executed each time the interrupt is triggered.
- 2) The main program, in the motion branch, that will include the code for using the TML interrupts.

3.1 Time period Interrupt routine

The "Interrupts" section allows you to customize the TML interrupt service routines. Once the "User defined" option is marked the interrupt routine will appear in the project window (left side), under the "Interrupts" section.

		Work	offline Set all a	apps offline	12.1	0	9	K X	A	Axis off
Application tree * #	latamata 🔒				- 1997 - 1994 -		23			100
P091.APN8 TML interrupt routines using	interrupts of									
TML interrupt routines - [1]	TML Interrupt	Default	User defined							
S Setup	int0 - STO/Enable input has changed	۲	0		Go to code					
Motion	int1 - Short-circuit	۲	0		Go to code					
- P Functions	Int2 - Software protections	۲	0		Go to code					
 Interrupts 	int3 - Control error	۲	0		Go to code					
int10 - Time period has elapsed	int4 - Communication error	۲	0		Go to code					
- 🛃 CAM Tables	Int5 - Position wraparound	۲	0		Go to code					
Memory settings	int6 - I.SP programmed transition detected	۲	0		Go to code					
	int7 - LSN programmed transition detected	۲	0		Go to code					
	int8 - Capture input transition detected	۲	0		Ga to code					
	int9 - Motion is completed / in velocity	۲	0	NEADER BETAKK	Go to code					
	Int10 - Time period has elapsed	0	۲	Reload default	Go to code					
	intil - Event set has occurred	۲	0		Go to code					
	int12 - Position trigger 14 change detected	۲	0		So to code					
	Int13 - Digital Input X programmed transition	۲	0		Go to code					

Figure 2 – Customizing the TML interrupt service routines

This application uses the "Int10 – Time period has elapsed" interrupt routine, to check the value of the "LED_status" user variable and command the OUT(0) digital output.

In order to customize the content of an interrupt service routine, it is necessary to click on the "Go to code" link. This will navigate to the specific section of the application where the desired functionality for the interrupt can be defined and implemented.

The "Jumps and Function Calls" dialogue allows to control the TML program flow through unconditional or conditional jumps and unconditional, conditional or cancelable calls of TML functions. In this application, the "Jumps and Function Calls" dialogue was used to create some conditional jumps, function of the "LED_status" user variable (if "LED_status = 0" the "OUT(0)" output is set to the active level, to switch ON the LED. Otherwise, the "OUT(0)" is set inactive, to switch OFF the LED).

PlasyMotion Studio II - P091.APN8 TML interrupt routil	nes using ntrol panels Utilities Options	Help							201	U	2
			Work offline	Set all apps offline	2 3		9	蕉	×		Axis E
pplication tree * #											-
P091.APN8 TML interrupt routines using		1ed						, ھ	K 4	+	1/4
TML interrupt routines - [3]	∧ S Curve	600	DIED ON LED	tatus CO- //Dranch to	OLED CONTR	LDD erstur	- 0	-		3	125
- S Setup	MPT	60	IO LED_ON, LED_S	status, ccc //branch to	TED_ON II	CED_Status a					dep 1
 M Motion 	N PVT	Jumps	and Function Cali	3		10 - 14 (P. N. D.				×	
- 🔂 Homings	Q− External	• Go	to	Address	s / Label LE	D_ON					1
- Punctions	유용 Electronic Gearing	O Call		🔽 lf	LED_status		27	. 0		0 1	V.
- 🔐 Interrupts	Electronic Camming	O Can	icelable call	R	Sec.			< 0		≤ 0	-
i int10 - Time period has elapsed	D Motor Commands							> 0		≥ 0	0
CAM Tables	🚀 Position Triggers	O. Ret	um from function	E		1					
MS Memory settings	Homing	O Ret	urn from interrup	t							P1
CP Control panels	TA Contouring	C Abo	ort cancelable call								0
1_Motion_Status	T/= Test	O Inst	ert label								1
- @ 2_Drive_IO	IO Events						OK	Cancel		Helo	0)
- @ 3_CANopen_Status	1 Jumps and Function Calls	ļ.,						concer	9.9	resp	0
- @ 4_CANopen_Objects	Inputs / outputs										-0
S_Drive_Status	16 16-bit Integer										-0
	32 32-bit Long or Fixed										-
	* Arithmetic										
	🔁 Data Transfer Between Axes	100									U ¹
	Send Data To Host										
	MISC Miscellaneous	-									

Figure 3 – Conditional jump to LED_ON label based on LED_status

The "I/O" dialogue allows different operations with the drive digital inputs and outputs. It was used here to set the "OUT(0)" digital output LOW or HIGH (function on the "LED_status" variable value). This way, the LED connected to this input is switched ON or OFF.

		Work offline Set all apps offline X X In In
cation tree 🔹 🤻 🕸		
P091.APN8 TML interrupt routines using	Trapezoidal Profiles	led ₽ × + →
TML interrupt routines - [3]	∧ S Curve	GOTO LED. ON, LED. status, EQ: //Branch to LED. ON if LED. status == 0
- S Setup	M PT	OUT(0) = 0x0001: //Set 1/O line 0 to High
 M Motion 	NO PVT	Invested and and a
- 🔀 Homings	Q− External	
- 🕑 Functions	유용 Electronic Gearing	○ Read input into variable
 Interrupts 	K Electronic Camming	Single I/O Set output OUT0 OUT0 F O Low High
int10 - Time period has elapsed	3 Motor Commands	Liset I/D line bine as output
CAM Tables	Position Triggers	
HS Memory settings	Homing	C Read input(s) into variable
CP Control panels	M Contouring	· · · · · · · · · · · · · · · · · · ·
- @ 1_Motion_Status	T/~ Test	○ Set output(s)
- 🐼 2_Drive_IO	IC* Events	
- @ 3_CANopen_Status	IJ Jumps and Function Calls	0000000000000000
- 🔄 4_CANopen_Objects	Inputs / outputs	
S_Drive_Status	16 bit Integer	S High / Low S with value of variable
	32 32-bit Long or Fixed	Limit switch inputs functionality
	* Arithmetic	O Default
	Data Transfer Between Axes	O General purpose input
	Send Data To Host	Enable input functionality
	MISC Miscellaneous	O Default
	🖆 Interrupt Settings	O General purpose input
	but Free Text	

Figure 4 – Setting the I/O line to HIGH

Once the OUT(0) digital output status is changed, the "Assignment and Data Transfer – 16 bit Integer Data" dialogue is used to modify the "LED_status" variable value. This has the purpose to indicate that the LED is ON ("LED_status = 1") or OFF ("LED_status = 0").

			1998
	🖶 🖷 📓 🛤 💈	🛓 🙎 🔽 Work offline Set all apps offline 🥕 🟦 🕨 🛛 💡 😤 🛪 🕼 🛄	ALX.
lication tree 🔹 🔹 🕷			
P091.APN8 TML interrupt routines using	/ Trapezoidal Profiles	Î led	1/4
TML interrupt routines - [3]	∧ S Curve	GOTO LED ON LED status FO: //Reanch to LED ON if LED status == 0	12
- S Setup	M PT	OUT(0) = 0x00011 //Set 1/0 line 0 to High	「你
A Motion	NO PVT		1 1
- 🔛 Homings	Ø⊷ External	LED_Status = 0	1
- 🕑 Functions	유용 Electronic Gearing	16-bit Integer	
🖌 🎲 Interrupts	Electronic Camming	Set 16-bit variable LED_status	
int10 - Time period has elapsed	D Motor Commands	With value / 16-bit variable / label 0	
CAM Tables	Position Triggers		
	Homing	O with O Program	
CP Control panels	M Contouring	EZROM increment the pointer variable	
- @ 1_Motion_Status	T/* Test		
② 2_Drive_IO	It Events	O With Part of 32-bit variable	
- @ 3_CANopen_Status	NJ Jumps and Function Calls	O With inverse () of unrights	
4_CANopen_Objects	Inputs / outputs	O with inverse (*) of variable	_
- 3 5_Drive_Status	18 16-bit Integer	O Using masks AND mask h OR mask h	
	32 32-bit Long or Fixed	Data	_
	* Arithmetic	O With checksum of data located in O Program memory between adresses	Tat
	Data Transfer Between Axes	© E2ROM	
	Rend Data To Host		-
	MISC Miscellaneous	Data memory concents located at address set in pointer variable	
	🖆 Interrupt Settings	C Set © Program With value / variable +	
	but Gross Tout	increment the pointer variable	

Figure 5 – Setting the LED_status value to 0

After the "OUT(0)" digital output state is changed and the "LED_status" variable is set accordingly, the program should exit from the interrupt. This is done using the "RETI;" (return from interrupt) instruction. It's located in the "Jumps and Function Calls" dialogue.

<u>Remark</u>: In the second case (when the LED is switched off), the "RETI" instruction is not added anymore because the EasyMotion Studio II automatically places one at the end of the interrupt routine.

Appleadori communication loois con	aroi paneis ounaes opuons r	iop					
		🛓 🗔 🗌 Work offline Set a	all apps offline 🧳 🔊	2 1 0	2 % 1	X 🖾 🖾	
ion tree 🔹 🔻 🕸							
091.APN8 TML interrupt routines using	✓ Trapezoidal Profiles	led				,₽ × + +	
] TML interrupt routines - [3]	∧ S Curve	- GOTO LED ON LED status	FO: //Branch to LED. OF	VifIED status == (1		
S Setup	M PT	OUT(0) - 0x0001: //Set 1/01	line 0 to High	10 000000000000000000000000000000000000			
M Motion	N PVT	160 status = 0	ine o to righ				
- 📴 Homings	Ø⊢ External	DET/: //Paturo from interes	•				
- 😰 Functions	State Electronic Gearing	AEN, // Neturn from interrup				16151	
🖌 🕎 Interrupts	Electronic Camming	Jumps and Function Calls				×	
int10 · Time period has elapsed	Motor Commands	O Go to	Address / Label				
CAM Tables	Position Triggers	Call	III II		18 I.	10	
MS Memory settings	🔂 Homing	 Cancelable call 					
CP Control panels	M Contouring						
- 🐼 1_Motion_Status	T/⊂ Test	Return from function					
- @ 2_Drive_IO	It Events	Return from interrupt					
- @ 3_CANopen_Status	N Jumps and Function Calls	O Abort cancelable call					
© 4_CANopen_Objects	Inputs / outputs	O Insert label					
S_Drive_Status	16 bit Integer			OK	Cancel	Help	
	32 32-bit Long or Fixed					2. cede	
	* Arithmetic						
	Data Transfer Between Axes						
	Send Data To Host						
	MSC Miscellaneous						
	🖌 Interrupt Settings						

Figure 6 – Return from interrupt

The LED_ON label is generated using the "Jumps and Function Calls" wizard and is used to indicate where the program should jump if the "LED_status == 0" condition is true.

		🛓 😡 🗌 Work offline Set	all apps offline 🧪	± + 0	🔮 💃 🗙	6 6
ation tree 🔹 🕷						
P091.APN8 TML interrupt routines using	Trapezoidal Profiles	led			,	B x ← →
TML interrupt routines - [3]	∧ S Curve	GOTO LED ON LED status	EQ: //Branch to LED C	N if LED status == (5	
- S Setup	M PT	OUT(0) = 0x0001: //Set I/O	line 0 to High			
 M Motion 	N PVT	LED status - 0	inc v tv righ			
- 🚯 Homings	Q− External	RET! //Return from internin	st.			
- 🔛 Functions	유용 Electronic Gearing	IED ON: //Defice label IED	ON			
 Interrupts 	Electronic Camming	CED_ON WDenne laber CED_				
int10 · Time period has elapsed	D Motor Commands	Jumps and Function Calls				×
CAM Tables	Position Triggers	O Go to	Address / Labe	4		
MS Memory settings	Homing	O Call	(iii) tt		+ 0	
CP Control panels	M Contouring	Cancelable call				
- I_Motion_Status	T∕~ Test					
- @ 2_Drive_IO	IC+ Events	Return from function				
- 💿 3_CANopen_Status	N Jumps and Function Calls	O Return from interrupt				
- @ 4_CANopen_Objects	Inputs / outputs	O Abort cancelable call				
- @ 5_Drive_Status	16 16-bit integer	Insert label LED_ON				
	32 32-bit Long or Fixed			OK	Cancel	Help
	* Arithmetic			0.0		AND STREET
	Data Transfer Between Axes					
	Send Data To Host					
	HISC Miscellaneous					
	🐓 Interrupt Settings					
	but Free Text					

Figure 7 – Defining LED_ON label

The remaining instructions from the ISR10 are complementary to what was presented already.

Each time the ISR10 gets executed the state of the LED will be toggled depending on the value of the LED_status variable.

3.2 Main motion section

The "Miscellaneous" dialogue allows to declare user variables, reset the drive/motor fault status, execute the "END" / "NOP" / "ENDINIT" TML instructions, change the CAN / RS-232 baudrate and save the actual setup into the drive memory. In this case the "Miscellaneous" dialogue was used to declare "LED_status" the 16bit integer user variable, that is used to track the output status (active or inactive).

ect Application Communication Tools Co	trol panels. Utilities Options Help		
		😨 🐼 Work offline Set all apps offline 🛛 者 👔 👘 0 🕴 🖡 🗙 🎫 🚳	
lication tree 🔹 🔻			
P091 APN8 TML interrupt routines using		Search Search	0
TML interrupt routines - [3]	∧ S Curve	- int FD status // Define integer variable LFD status	1
Setup	M PT		
 M Motion 	10 PVT	Miscellaneous X	
- 📴 Homings	0- External	Define variable int LED_status	
- 🕑 Functions	유용 Electronic Gearing	O Reset FAULT status	
🖌 🔃 Interrupts	Electronic Camming	Insert END instruction	
int10 - Time period has elapsed	Motor Commands	O Insert NOP instruction	
CAM Tables	Position Triggers	O faile second and the balance	
- MS Memory settings	Homing	Sense communication - Set baugrate	1
CP Control panels	M Contouring	CAN communication - Set baudrate	
- 💿 1_Motion_Status	T/⁻ Test	O Insert ENDINIT instruction	,
- 💿 2_Drive_IO	Ir• Events	 Save actual setup data in EEPROM 	E
- 🐼 3_CANopen_Status	NJ Jumps and Function Calls	OK Cancel Help	I
4_CANopen_Objects	Inputs / outputs		+
- 💿 S_Drive_Status	16 bit Integer		ŧ
	32 32-bit Long or Fixed		1
	* Arithmetic		5
	To Data Transfer Between Axes		
	Send Data To Host		
	HISC Miscellaneous		
	🖆 Interrupt Settings		
	bet Free Text 👻		

Figure 8 – Defining variable LED_status

The "Assignment and Data Transfer – 16 bit Integer Data" dialogue allows different operations with the 16-bit integer variables / parameters / registers. Here it was used to initialize the "LED_status" user variable with 0.

oject Application Communication Tools Cor	ntrol panels Utilities Options	Help	
		🟦 📮 🗹 Work offline Set all apps offline 🥻 🟦 🕨 😐 🖉 🍯 👘 😤 🕺 🐻	
oplication tree 🔹 🔹 🕸			
P091.APN8 TML interrupt routines using	Trapezoidal Profiles	Search Start	0/0
TML interrupt routines - [3]	∧ S Curve	int LED_status; // Define interer variable LED_status;	123
Setup	M PT	IED come = 0	1 dep
 M Motion 	NO PVT	ELECTION OF THE	
- Mornings	€ External	16-bit Integer	
- 😥 Functions	유용 Electronic Gearing	Set 16-bit variable LED_status	
🖌 🌇 Interrupts	Electronic Camming	With value / 16-bit variable / label 0	
🗆 💽 int10 - Time period has elapsed	D Motor Commands	(2) Data	
CAM Tables	Position Triggers	O With Program	
HS Memory settings	Homing	E2ROM increment the pointer variable	
CP Control panels	M Contouring	O Low	
- @ 1_Motion_Status	T∕~ Test	With Part of 32-bit variable	
Q 2_Drive_IO	It Events	With inverse (-) of variable	
- 🔅 3_CANopen_Status	11 Jumps and Function Calls		
- @ 4_CANopen_Objects	Inputs / outputs	O Using masks AND mask: h OR mask	
S_Drive_Status	16 -bit Integer	Ø Data	
	32 32-bit Long or Fixed	O With checksum of data located in O Program memory between adresses	h
	* Arithmetic	C EZROM	
	🔁 Data Transfer Between Axes	The STOP Company of the Antonia State Stat	
	Rend Data To Host	O Data Memory concerns sociality and address set in pointer values	
	MISE Miscellaneous	(b) set to Program With value / variable	
	🚽 Interrupt Settings	increment the pointer variable	
	bd Free Text	OK Cance	E F

Figure 9 – Setting the LED_status value to 0

The "Interrupt Settings" dialogue allows to activate and/or deactivate the TML (Technosoft Motion Language) interrupts. In this case, it was used to activate the "int10 – Time period has elapsed" interrupt routine and set it to a time period of 0.5 s.

		Set all apps offline	2		0 9 1	X 6 8
on tree 🔹 🖷						
91.APN8 TML interrupt routines using		Search				
TML interrupt routines - [3]	∧ S Curve	int LED, status // Define integer unishin LED,	and the second se			
S Setup	M PT	The cell status of beine meger variable ceb.s	status			
M Motion	N PVT	LED_Status = 0;	1000000000	witz		
🛞 Homings	0 ← External	TMUNTPER = 5000;//Set a time interrupt at en	very 0.5[s	ster }		
- P Functions	Se Electronic Gearing	SRB ICR, 0x8FFF, 0x0400; //Set/Reset Bits of In	terrupt G	ontrol Re	gister	
Interrupts	12 Electronic Camming	EIN1; //Enable TML INTerrupts				_
int10 - Time period has elapsed	3 Motor Commands	Interrupt Settings				×
CAM Tables	Position Triggers	TML Interrupts				
MS Memory settings	🔂 Homing	Enable globally Disable globally				
CP Control panels	M Contouring		2 23		Reset previous TML	la contra l
- @ 1_Motion_Status	T/~ Test	I'ML Interrupt	Enable	Disable	interrupt requests	Details
- @ 2 Drive IO	Ir* Events	Int0 - STO/Enable input has changed				
- @ 3 CANopen Status	N Jumps and Function Calls	Int1 - Short-circuit				
- @ 4 CANopen Objects	Inputs / outputs	Int2 - Software protections				81
- 🐼 5 Drive Status	16 16-bit Integer	Int3 - Control error				
	32 32-bit Long or Fixed	Inte - Communication error			1	
	* Arithmetic	Into - Position wraparound		-		
	2 Data Transfer Between Axes	Int7 - LSN programmed transition detected				95
	Send Data To Host	Int8 - Capture input transition detected				
	MISC Miscellaneous	Int9 - Motion is completed / in velocity	n			
	Interrupt Settings	Inti0 - Time period has elapsed			v	=
	but Free Text	Time period 0.5 s +				
		Int11 - Event set has occurred			U 1	
		Int12 - Position trigger 14 change detected				
		Int13 - Digital Input X programmed transition				
					OK Cancel	Help

Figure 10 – Interrupt settings

4. Application evaluation

After running the TML program, we can use the "2_Drive IO" control panel in EasyMotion Studio II to monitor the status of the OUT0 digital output.



Figure 11 – Using the "2_Drive IO" control panel

To start a control panel, right-click on it, and then select "START" from the context menu that appears.



Figure 12 - Starting the "2_Drive IO" control panel