

Application Note

AN2353/D
Rev. 0, 10/2002

eTPU Programming
Essentials: Architecture
Overview



Mike Pauwels &
Jeff Loeliger

This document is first in a series of application notes whose purpose is to help designer determine applicability of the enhanced Time Processor Unit (eTPU) for particular applications, understand setup and programming criteria, and the input/output timing module available on Motorola microcontrollers. The eTPU is a new generation of programmable peripheral modules suitable for generating and detecting complex signals with minimal direct support from the main processor.

1 Overview

The Time Processor Unit (TPU) has been a highly successful peripheral timer module. It has enjoyed wide acceptance and acclaim despite a very complex architecture, limited tools support, and serious limitations in its ability to measure or drive very fast signals.

The enhanced TPU was designed to address limitations of the TPU. It is not simply an extension of the original, but a significant redesign based on the successful TPU architecture (see Table 1). While TPU code will not run without modification on an eTPU, there is no function that cannot be ported. Furthermore, most of the known application requirements that were out of the reach of the TPU can be easily met by the eTPU.

Table 1. Feature Comparison: TPU vs. eTPU

	TPU	eTPU	eTPU Comments
Channels	16	32	Separate input & output
Channel Modes	2	13+	See channel details
Time Bases	2	2 of 4	Shared between all timer modules
Parameters	200-256 bytes	2-16 Kbytes	Defined per MCU
Code Memory	2-8 Kbytes	6-64 Kbytes	Shared between two eTPU engines
Fastest Thread	12 clocks	8 clocks	Max clock speed greater than 2x TPU
Register width	16 bits	24 bits	Top parm byte is accessible
Angle Clock	Software	Yes	Software supported hardware
16x16 Multiply	34 clocks	6 clocks	Also DIV and MAC
Compiler	No	ANSI C	3rd party source
Debugger	3rd Party	Yes	Multicore Nexus debugging
Simulator	3rd Party	3rd Party	Also integrated into CodeWarrior

Channel Hardware

The TPU was typically offered in multiple units on a single microcontroller. Functions had to be partitioned carefully, because there was no common data storage, timing, or execution links between the modules. The eTPU, on the other hand, is typically configured in an array of up to 64 channels and two engines sharing a common instruction and parameter memory and debug interface. Timer buses can be shared and any of the 64 channels can signal any other. Mechanisms are provided for coherent data transfers between channels as well as between the CPU and the eTPUs.

The eTPU boasts several major areas of improvement over the TPU, including significant changes in the channel hardware, the memory, the microengine, and the tools. Many of these improvements are outlined in subsequent sections and each will be examined in detail in future application notes.

2 Channel Hardware

The TPU provided 16 identical channels, each providing:

- Capture Register: Can be configured to latch one of two sixteen bit counter buses on a pin transition
- Compare Register: Can force a pin transition on a programmed match with one of the buses.

The match or capture events could be used to request service by the microengine, which could then reload the registers for the next event. While the timing on a single event could be accurately determined by this hardware, the minimum setup time for the second event was determined by system latency as well as by the software service time for the first event. This latency limited the minimum guaranteed time between pin actions.

Each eTPU channel provides two compare and two capture registers. With the modes selected properly, it is possible to generate or measure pulses as narrow as one clock cycle. Registers and buses are 24 bits wide, a fact which greatly extends the dynamic range of any function. One of the buses can be driven by a special Angle Clock subsystem, allowing pulses to be started or stopped in various combinations of the time and angle.

The eTPU channel hardware can be configured to at least thirteen distinct modes to meet a large number of application requirements. For each of these modes, match registers and capture registers can each be connected to different timer-counter registers, providing another degree of flexibility. Finally, in some microcontrollers, the input function of the channel may be internally connected to a different pin from the output function. These multiple alternatives provide an extensive array of potential channel hardware configurations. Some of the more interesting configurations are:

- Single input capture or single output compare provides the original TPU functionality and should be the starting point for any design.
- Input transitions can be made to capture both the time and angle of the transition.
- Input capture can be conditioned on combinations of matches. This provides the capability of finding an expected pin transition in a window of time, angle, or combination matches.
- Multiple input captures can be used to detect and measure very small pulses.
- One match blocked by another, such as a drive signal triggering on a crank shaft angle which is inhibited by a timeout.
- One or two matches blocked by a transition.
- Output disable modes which can disable output drives immediately when an input signal is detected outside of a predetermined window. This mode could protect an output drive circuit in the event a short circuit is detected.

3 Memory

When the TPU was first designed, it was seen as a somewhat oversized timer in a very large microcontroller. The concept of multiple processing units on a single chip was untried in the market, so every effort was made to keep the footprint of the module as small as possible. The TPU on the MC68332 was introduced with just 2 Kbytes of code memory which was shared between 32 bit microinstructions and 16 bit entry point vectors. The parameter space was 200 bytes arranged as 100 parameters distributed among the 16 channels as 6 or 8 local parameters for each channel, with a provision for any channel to address memory allocated to a different channel. A few years after the original design, the TPU2 introduced a paging scheme for the program memory, effectively raising the maximum limit to 8 Kbytes; and stretched the parameter space to 256 bytes. Still, memory size was the factor that most often limited applications.

The eTPU is designed with an addressable program space of up to 16K microinstructions (64 Kbytes) which can be shared between two microengines. A particular function can be run on both microengines simultaneously without any degradation of performance. In addition, each channel can now address up to 128 local parameters (512 bytes) and 256 (2 Kbytes) in global address space, limited only by the parameter RAM available on a given MCU. The local parameter space for each channel is determined by the host at system initialization, providing the most efficient distribution of the available parameters. A function such as Pulse Width Modulation may be allocated two control parameters, while another such as a stepper motor drive may require dozens. Provision has been made to enable DMA access of the parameter RAM to greatly increase the virtual size of the data memory.

The parameters are 32-bits wide to match the bus width of the core processor. Since the eTPU is a 24-bit machine, various means are provided for efficient transfer of parameter information, including sign extension and separate accessing of the most significant byte. Transfers between the eTPU and the host can be protected for coherency by use of hardware supported semaphores.

4 The Microengine

The TPU was designed to service the channel timer hardware as efficiently as possible, by providing a small, microcoded instruction map, see Figure 1, which could perform several operations in parallel. There were five microcode formats with as many as twelve operations which could be executed in parallel. All microinstructions executed in a single microcycle (2 clock cycles) except where there was a collision on memory access between the CPU and eTPU.

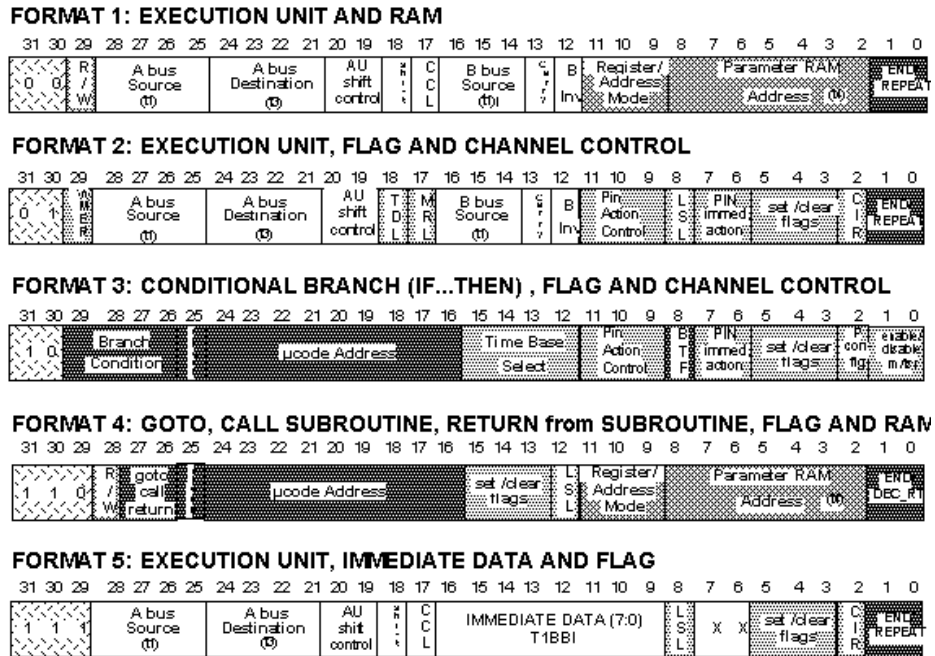


Figure 1. TPU Instruction Set Coding

Because of more complex options in the greatly expanded channel hardware, the eTPU instruction map, see Figure 2, is not nearly as simple and small, but many instruction combinations can still be done in a single cycle. Familiar routines where a microengine stores information latched by the channel hardware and the channel is setup for the next service can still be done in a few microinstructions. Multiply and Divide functions, which may take multiple cycles, can be executed in parallel with unrelated ALU functions.

format	microinstruction																																					
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0						
A1	000			IMM[15:13]				IMM[7:2]				IMM[23:16]						IMM[12]	RTN	IMM[11:9]			IMM[1:0]		T2D		IMM[8]		00									
A2	000			IMM[15:13]				IMM[7:2]				IMM[23:16]						IMM[12]	CCS	IMM[11:9]			IMM[1:0]		ABSE	ABDE	IMM[8]		01									
A3	000			ALUOP[4]		CCSV		IMM[7:2]				IMM[23:16]						ALUOP[3:2]		AS/CE			IMM[1:0]		ALUOP[1:0]		0											
A4	000			FLC[2]		CCSV		IMM[7:2]				IMM[23:16]						CCS		FLC[1:0]			IMM[1:0]		ABSE	ABDE	1											
B1	10		0	SHF		IMM[7:2]				IMM[23:16]						SRC		REP			AID[7:0] (global param)																	
B2	10		1	SHF		IMM[7:2]				IMM[23:16]						SRC		P/D			CCS			ZRO			AID[6:0] (channel param)											
B3	000			END		IMM[7:2]				IMM[23:16]						RW		T4ABS			T2ABD			STC		ABSE	ABDE	rsv	11									
B4	001			0		CIN		BINV		T4BBS				IMM[23:16]						1		AS/CE			SMPR		ALUOP											
B5	001			0		CIN		BINV		T4BBS				IMM[23:16]						0		AS/CE			SMPR		ALUOP											
B6	001			1		CIN		BINV		T4BBS				IMM[23:16]						rsv		AS/CE			SMPR		ALUOP											
B7	011			END		SHF		IMM[7:2]				IMM[23:16]						TDL		PSC			MRL1		ERW1		MRL2		ERW2		ABSE	ABDE	CCS	MRL	PSCS			
C1	010			0		OPAC1		OPAC2		TBS1				TBS2						LSR		PSC			MRL1		ERW1		MRL2		ERW2		PDCM					
C2	010			1		IPAC1		IPAC2		TBS1				TBS2						LSR		PSC			MRL1		ERW1		MRL2		ERW2		PDCM					
D1	110			0		MRL		PTC		PSC				FLS		RW		PSCS		CIRC			R/D		AID[7:0] (global param)													
D2	110			0		MRL		PTC		PSC				FLS		RW		PSCS		CIRC			R/D		AID[6:0] (channel param)													
D3	111			rsv		MRL		PTC		PSC				FLS		FL		PSCS		CIRC			R/D		STC		11		00		rsv							
D4	111			rsv		MRL		PTC		PSC				FLS		FL		PSCS		CIRC			R/D		SMPR		11		00		rsv							
D5	110			1		MRL		PTC		MTD				CCM		RW		TDL		FLC			MRL1		ERW1		MRL2		ERW2		AID[7:0] (global param)							
D6	110			1		MRL		PTC		MTD				CCM		RW		TDL		FLC			MRL1		ERW1		MRL2		ERW2		AID[6:0] (channel param)							
D7	111			rsv		MRL		PTC		MTD				CCM		RW		TDL		FLC			MRL1		ERW1		MRL2		ERW2		STC		11		01		rsv	
D8	111			rsv		MRL		PTC		MTD				CCM		RW		TDL		FLC			MRL1		ERW1		MRL2		ERW2		SMPR		11		01		rsv	
E1	111			rsv		J/C		BCC				FLS		RW		BCF		BAF[13:0]						00		P/D		STC										
E2	111			rsv		J/C		BCC				FLS		RW		BCF		BAF[13:0]						01		AID[2:0]		rsv										
E3	111			rsv		J/C		BCC				FLS		RW		BCF		BAF[13:0]						10		rsv		SMPR		rsv								
E4	111			rsv		J/C		BCC				FLS		RW		BCF		BAF[13:0]						11		1		rsv		rsv								
F1	111			rsv		J/C		BCC				FLS		RW		BCF		BAF[13:0]						111		rsv		rsv		rsv								
format	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0						

Figure 2. eTPU Instruction Set Coding

The Microengine

The number of sources that can request service for a given channel has been increased, and the entry vector table for each function has been doubled from 16 to 32. In addition, an alternate entry table can be selected offering still more flexibility. Since many applications are designed with large and complicated state tables, a Dispatch function has been provided to more quickly vector to the correct execution thread.

The comparative programming models for the TPU and eTPU are shown in Figure 3.

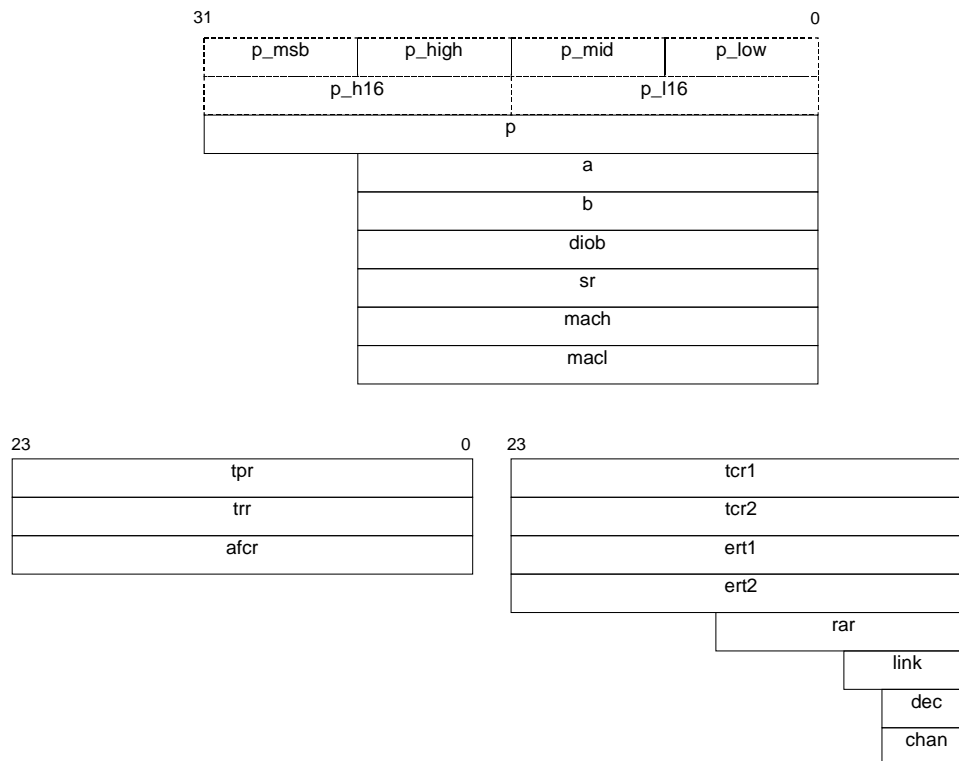


Figure 3. Comparative Programming Models: TPU (left) and eTPU

Performance of the microengine has been enhanced by stretching the registers and arithmetic unit to 24 bits and extending the maximum clock speed of the eTPU. In addition, the 10 clock Time Slot Transition of the TPU has been reduced to 6 clocks. The new instruction set includes some badly needed bit manipulation instructions. The eTPU also boasts a Multiply and Divide unit which can even execute a MAC instruction. Despite the great increase in complexity, initial studies suggests that automotive functions which required multiple word parameters can be programmed more efficiently on the eTPU than the TPU.

The eTPU includes hardware support for an Angle Clock function. This is a feature that inserts a selected number of subdivisions between periodic input pulses such as might be produced by a toothed wheel sensor. Features of the Angle Clock include a selectable tick rate, missing tooth corrections, and full monotonic position counts. The angle clock register can be used as a compare base, enabling any function to operate in the time domain, the angle domain, or both.

Time and angle counter buses can be exported to additional eTPUs or other compatible modules such as the Enhanced Modular I/O System (eMIOS). All channels of the timing system can operate with a common time and angle base.

A debug mode, internal register visibility, and Nexus Class 3 interface provide a much greater level of silicon development support than that of the TPU.

5 Tools

The TPU was never expected to be programmed by customers and hence the tool and application support fell short of market expectations. In spite of this the TPU steadily gained acceptance thanks to some extraordinary programmers, consultants, and independent tool vendors who persevered despite the minimal support.

With the eTPU, tools are being developed with the silicon and will be generally available in advance of the first eTPU microcontroller. Included in the Motorola sponsored tool set are an ANSI C compiler from Byte Craft, a cycle accurate eTPU simulator from ASH WARE, and a source level debugger from Metrowerks. In addition, Metrowerks has integrated all these tools with the core development and simulation tools into the popular CodeWarrior IDE to support the MPC5500 series of eTPU-based microcontrollers from Motorola.

The eTPU compiler will support an ANSI compliant source, which will provide a degree of portability to future compatible devices. The eTPU source files are designed to automate memory allocation, resolve variable references, and simplify host-eTPU interfaces. The C language access to the eTPU will not make an eTPU programmer out of any C coder, but it will make the eTPU much more accessible to a real-time systems engineer. With modern optimization technology, we expect the compiler to compete reasonably with well-written assembly code.

The compiler will emit Elf/Dwarf format files making possible source-level debugging of the eTPU microcode. Metrowerks offers a full-feature eTPU debugger integrated in their popular CodeWarrior IDE.

Real-time controller debugging is difficult at best and often impossible at full speed. As part of the effort to provide the necessary development support for the eTPU, Motorola has engaged ASH WARE to extend their popular TPU Simulator to support the eTPU. This tool is being co-verified against the silicon design to ensure accurate behavior of the model as well as correct operation of the silicon to the system requirements.

The standalone tools can provide adequate support for developing independent functions on the eTPU. Modern systems, however, are moving toward true distributed processor control systems. For example, while the eTPU is capable of gathering raw data at a very high rate, many systems will require the data to be processed in the main CPU before closing the loop to an output device.

A transaction accurate, full chip simulator provided by Motorola will provide simulation of the multicore controller interfaced to a Metrowerks multicore debugger front end. Critical elements of a design can be checked for logical correctness and timing compatibility before running on silicon. When the silicon target is available, the same debugger interface will be available to trace the operation through Nexus.

Additional applications support for the eTPU will include sample functions and initialization code, user manuals, application notes, training courses, and direct sales and applications help.

6 Summary

Whether an application is simply a faster version of an older real-time function or a new software emulation of an expensive and inflexible ASIC, the enhancements found in the eTPU have greatly increased the capabilities of the Time Processor Unit to address any requirements. Applications such as spark refiring, complex stepper motor acceleration, or AC induction motor control are now within easy grasp the eTPU. The larger memory and the enhanced microengine can simplify a number of complex TPU applications, while the tools will make programming the device significantly easier for the real-time system engineer.

HOW TO REACH US:**USA/EUROPE/LOCATIONS NOT LISTED:**

Motorola Literature Distribution
P.O. Box 5405, Denver, Colorado 80217
1-303-675-2140 or 1-800-441-2447

JAPAN:

Motorola Japan Ltd.
SPS, Technical Information Center
3-20-1, Minami-Azabu Minato-ku
Tokyo 106-8573 Japan
81-3-3440-3569

ASIA/PACIFIC:

Motorola Semiconductors H.K. Ltd.
Silicon Harbour Centre, 2 Dai King Street
Tai Po Industrial Estate, Tai Po, N.T., Hong Kong
852-26668334

TECHNICAL INFORMATION CENTER:

1-800-521-6274

HOME PAGE:

<http://www.motorola.com/semiconductors>

Information in this document is provided solely to enable system and software implementers to use Motorola products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Motorola reserves the right to make changes without further notice to any products herein.

Motorola makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Motorola assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters which may be provided in Motorola data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. Motorola does not convey any license under its patent rights nor the rights of others. Motorola products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Motorola product could create a situation where personal injury or death may occur. Should Buyer purchase or use Motorola products for any such unintended or unauthorized application, Buyer shall indemnify and hold Motorola and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Motorola was negligent regarding the design or manufacture of the part.



Motorola and the Stylized M Logo are registered in the U.S. Patent and Trademark Office. digital dna is a trademark of Motorola, Inc. All other product or service names are the property of their respective owners. Motorola, Inc. is an Equal Opportunity/Affirmative Action Employer.

© Motorola, Inc. 2002