Freescale Semiconductor
Application Note

AN2353
Rev. 1, 08/2004

The Essentials of Enhanced Time Processing Unit

by: Mike Pauwels & Jeff Loeliger
TSPG

1 Purpose

The purpose of this Application Note is to help the designer understand the basic structure of the Enhanced Time Processing Unit (eTPU), an Input/Output timing module available on Freescale 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.

This document is an overview of the architecture designed to help the engineer decide whether the eTPU is suitable for a particular application. There is an emphasis on the features of the eTPU that were enhanced to address limitations of the TPU (and TPU2/TPU3) design, with notes on how those features can be used to address a specific problem.

2 Overview

The Time Processor Unit (TPU) has been the most successful peripheral timer module ever seen on a

Table of Contents

microcontroller. This is in spite of a difficult architecture, less than ideal tools support, and serious limitations in its ability to measure or drive very fast signals.

The Enhanced TPU was designed from the start to address the limitations of the TPU. It is not simply an extension of the original, but a significant redesign based on that successful architecture (see chart below). While TPU code will not run without modification on an eTPU, there is no function that cannot be ported. Meanwhile, most of the known applications 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 per MCU
Channel Modes	2	13+	See channel details
Time Bases	2	2 of up to 10	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 Defined per MCU
Fastest Thread	12 clocks	8 clocks	Max clock speed greater than 2x TPU
Register width	16 bits	24 bits	Top parameter byte is accessible
Angle Clock	Software	Yes	Software supported hardware
16x16 Multiply	34 clocks	6 clocks	Also DIV and MAC
Compiler	No	ISO C	3rd party source
Debugger	3rd Party	Yes	Multicore Nexus debugging Depends on the device
Simulator	3rd Party	3rd Party	Also integrated into CodeWarrior Depends on the device

The TPU was often made available in multiple units on a single microcontroller. Functions on the TPUs had to be partitioned carefully, as there was no common data storage, timing, nor execution between the modules. The eTPU is typically configured in an array of up to 64 channels, with two engines sharing a common instruction memory, data 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 improvements over the TPU in several major areas: the channel hardware, the memory, the microengine, and the tools. Many of these improvements are outlined below, and each will be examined in detail in future notes.

3 Channel Hardware

The TPU provided 16 identical channels each providing a capture register, which could be configured to latch one of two 16-bit counter buses on a pin transition; and a compare register, which could force a pin transition on a programmed match with one of the timer 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 a 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 timebase count. Registers and buses are 24 bits wide, which greatly extends the dynamic range of any function. A special Angle Clock subsystem can drive one of the buses, allowing pulses to start or stop in various combinations of the time and angle.

The eTPU channel hardware can be configured to at least 13 distinct modes to meet a large number of applications 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 that 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.

4 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 size 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 six or eight 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 stretched the parameter space to 256 bytes and introduced a paging scheme for the program memory, effectively raising the maximum limit to 8 Kbytes. 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 shared global parameters (2 Kbytes). The channels can also use indexed addressing to access data anywhere in data RAM of up to 16k. 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 data 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 using hardware supported semaphores.

5 The Microengine

The TPU was designed to service the channel timer hardware as efficiently as possible, by providing a small, microcoded instruction map that could perform several operations in parallel. There were five microcode formats with as many as 12 operations that 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.

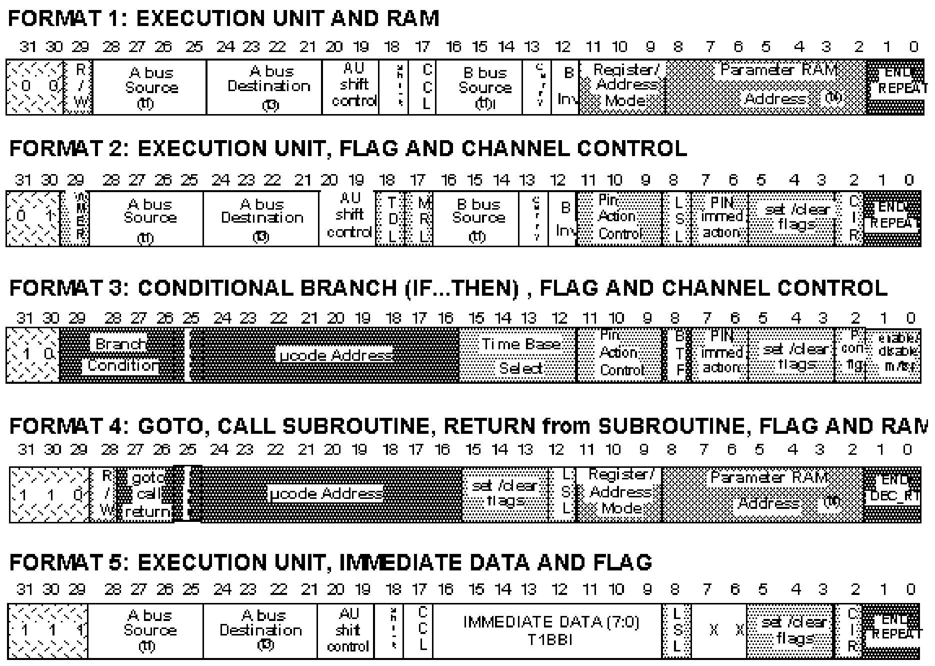

Figure 1. TPU Instruction Set Coding

Because of more complex options in the greatly expanded channel hardware, the eTPU instruction map 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 that may take multiple cycles to execute can be done in parallel with unrelated ALU functions.

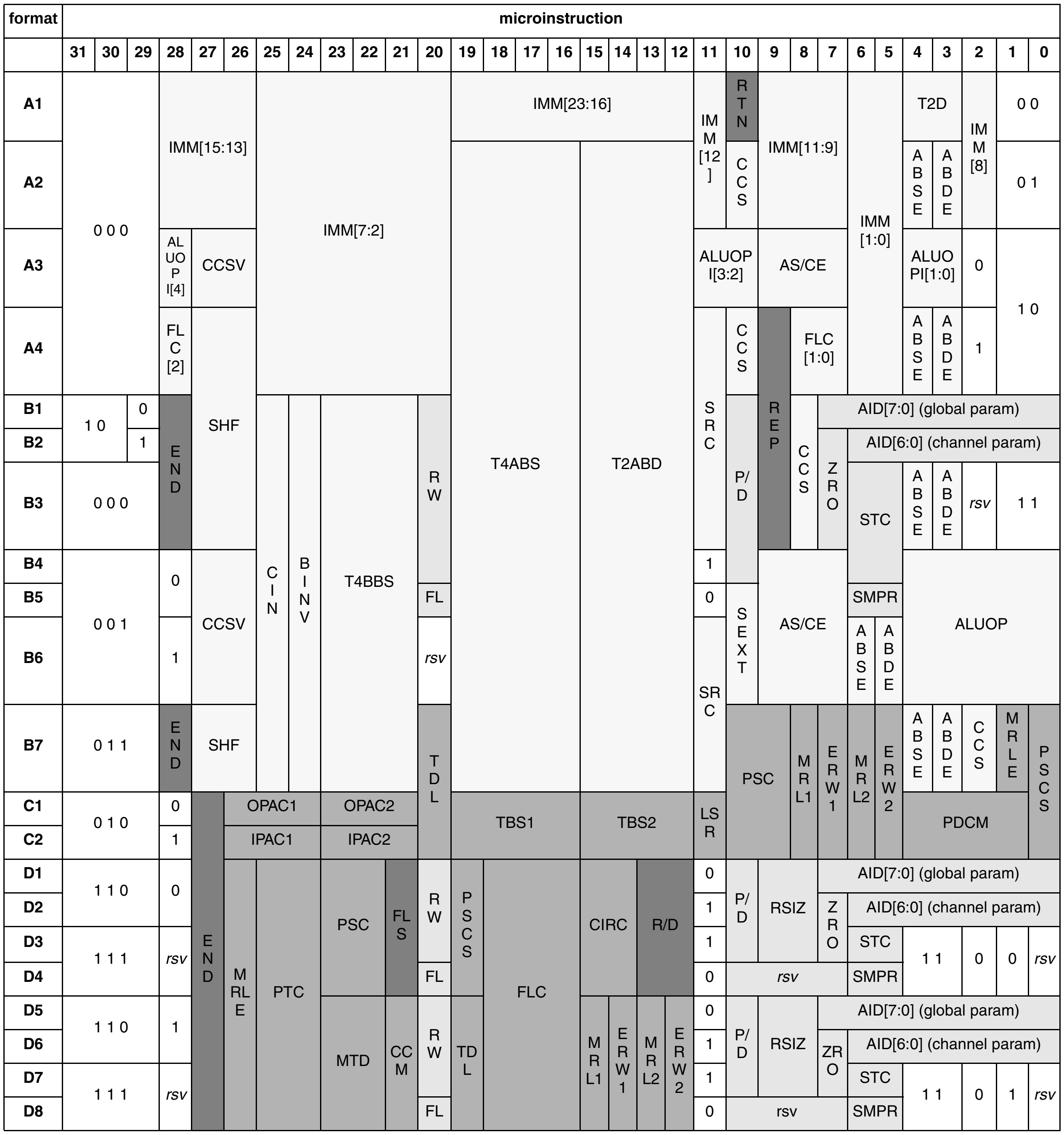

Figure 2. eTPU Instruction Set Coding

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
E1	1 1 1			*rsv*	J/C	BCC					FLS	RW	BCF	BAF[13:0]														00		P/D	STC	
E2																												01		AID[2:0]		
E3												FL																10		*rsv*	SMPR	
E4												0																11		1	*rsv*	
F1				*rsv*								1	*rsv*															111			*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 comparative programmer's models for the TPU and eTPU are shown in the figure.

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 much needed bit manipulation instructions. The eTPU also boasts a Multiply and Divide unit that can even execute a MAC instruction. Despite the great increase in complexity, a survey suggests that automotive functions that require multiple word parameters can be programmed more efficiently on the eTPU than the TPU.

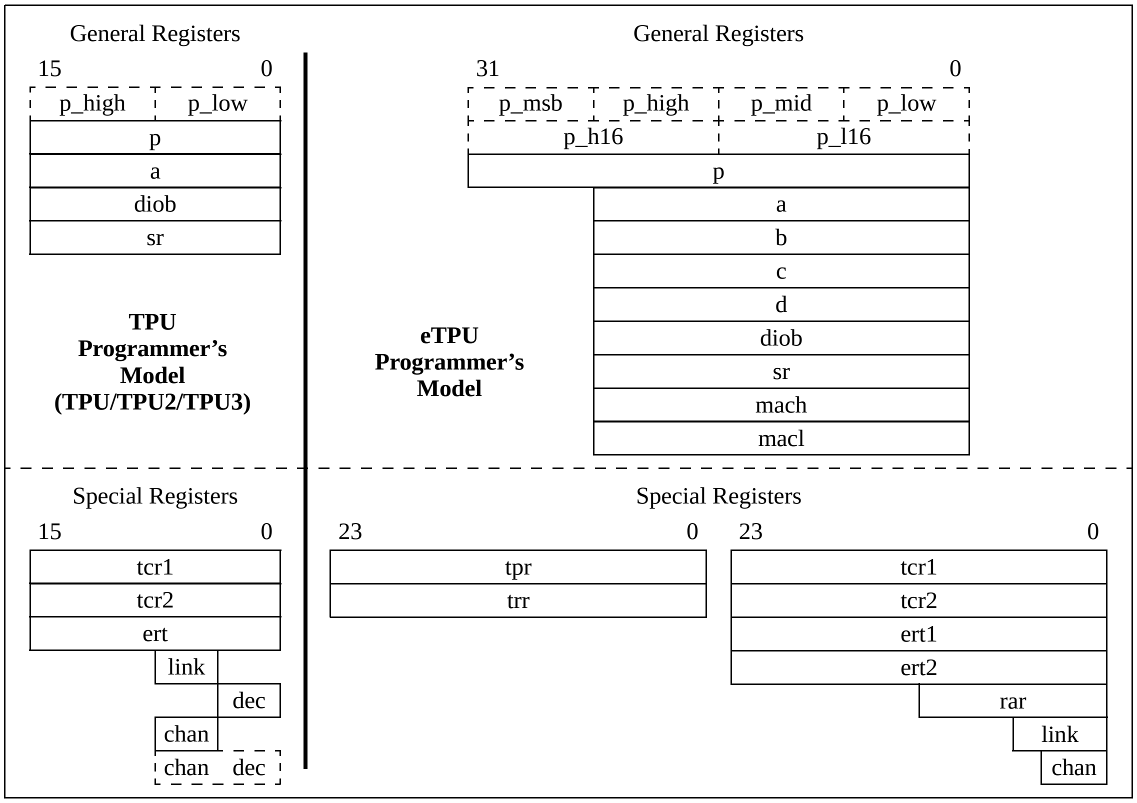

Figure 3. Comparative Programming Models: TPU (Left) and eTPU

The number of events 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 even more flexibility. This allows the eTPU to determine which thread needs to be run without needing to execute code. Since many applications are designed with large and complicated state tables, a new Dispatch instruction has been provided to more quickly vector to the correct execution thread.

The eTPU includes hardware support for an Angle Clock Function. This feature can be used to track an input pulse signal from a toothed wheel sensor and insert a number of equally spaced counts between each pulse to provide a counter that tracks the angle of the wheel. The counter can be used to sense inputs and drive outputs based on the angle of the wheel shaft. Features of the Angle Clock include a selectable tick rate, missing tooth corrections, and full monotonic position counts. The angle clock reference 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 reference.

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

6 Tools

The TPU was never expected to be programmed by customers; 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 tools vendors who persevered despite the minimal support.

With the eTPU, tools are being developed with the silicon and have been available in advance of the first eTPU microcontroller. Included in the Freescale sponsored tool set are an ISO C compiler from Byte Craft, a cycle accurate eTPU simulator from ASH WARE, and source level debuggers from Lauterbach and Metrowerks. In addition, Metrowerks has integrated all these tools with the core development and simulation tools into the popular Code Warrior IDE to support the MPC5554, the first eTPU microcontroller device from Freescale.

The eTPU compiler will support C source code compliant with the draft report on C for Embedded Systems, and will provide portability to future compatible devices. The eTPU source files will 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 and Lauterbach offer a full feature eTPU debuggers integrated into their popular IDEs.

Real-time controller debugging is difficult at best and often impossible at full speed. To provide the necessary development support for the eTPU, Freescale has engaged Ash Ware to extend their popular TPU Simulator to support the eTPU. This tool has been 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 stand alone 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.

Additional applications support for the eTPU will include sample functions, standard function sets (general, automotive and motor control), initialization code, manuals, application notes, and training courses.

7 Summary

Whether your application is simply a faster version of last year’s real time function, or a new software emulation of an expensive and inflexible ASIC, you will find that the enhancements of the eTPU have greatly increased the capabilities of the Time Processor Unit to address your requirements. Applications such as spark refiring, complex stepper motor acceleration, and 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.

THIS PAGE INTENTIONALLY LEFT BLANK

THIS PAGE INTENTIONALLY LEFT BLANK

THIS PAGE INTENTIONALLY LEFT BLANK

AN2353
Rev. 1, 08/2004