Welcome to this course on the Enhanced Time Processing Unit or eTPU. This module describes what an eTPU is and the benefits it offers to your designs. You will learn about the various applications and the associated library functions available for the eTPU and how easy it is to program using available graphical tools. You will also learn how to identify and access the various resources available to help you.

Course Introduction

Purpose:
This course describes what an eTPU is and how it can benefit your designs.

Objectives:
• Gain a basic understanding of the eTPU
• Learn about the eTPU functions and their structure
• Learn about the various eTPU tools that are available

Content:
20 pages
3 questions

Learning Time:
30 minutes

Because of an order from the United States International Trade Commission, BGA-packaged product lines and part numbers indicated here currently are not available from Freescale for import or sale in the United States prior to September 2010: MCF523X Products in 196 and 256 MAPBGA Packages
The enhanced time processor unit is a programmable I/O controller with its own core and memory system, which allow it to perform complex timing and I/O management independently of the CPU. The eTPU is essentially a self-contained microcontroller!

The best way to think of the eTPU is as a self-contained microcontroller dedicated to I/O functions. In other words, it's a microcontroller.

It consists of a CPU, which we call the micro-engine.

There is a hardware timer, which is shown as timer channels.

There is program memory or code memory for the program.

There is also dual ported data memory for all of the parameters and data. This is used to communicate with the host CPU in the rest of the system.

There is also a scheduler, which is like a small real time operating system implemented completely in hardware. The scheduler ensures that all of the service requests get processed in a timely fashion.

There is also a debug interface available through hardware.
## What Can Be Done With The eTPU?

<table>
<thead>
<tr>
<th>Category</th>
<th>Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>General Timing Functions</td>
<td>Input Capture, Output Compare, Pulse Accumulate, PWM</td>
</tr>
<tr>
<td>Serial Communications</td>
<td>UARTs, I2C, ARINC, Proprietary Protocols</td>
</tr>
<tr>
<td>Motor Control</td>
<td>Factory Automation, Robotics, Stepper Motor</td>
</tr>
<tr>
<td>Custom Logic Replacement</td>
<td>FPGAs and/or ASICs used for I/O</td>
</tr>
<tr>
<td>Engine Control</td>
<td>Spark Timing, Fuel Injection</td>
</tr>
</tbody>
</table>

Since the eTPU is a co-processor designed for timing control, it can handle complex timing functions. So, it can be used to implement general timing functions such as Input Capture, Output Compare, Pulse Accumulate, and PWM.

The eTPU can be used for various serial communications, specifically legacy type protocols such as UART, I2C, SPI. It can also be used for industry-specific protocols such as the ARINC protocol for aerospace.

It is also suitable for controlling DC motors, AC motors, and stepper motors. This also makes it very suitable for robotics or equipment with many moving parts.

The eTPU can also be used for more advanced tasks, such as replacing logic in the I/O. The typical guideline is that if the logic is in the lower speed I/O system then it is likely that the eTPU can process that logic. However, the eTPU is not capable of address decoding if the logic or FPGA is directly connected to the bus.

Additionally, the eTPU can also be used for engine control. It was initially designed specifically for the most demanding real-time control application today, which is internal combustion engine control. In fact, the eTPU is capable of running an engine under steady state type conditions with no servicing from the main CPU.
The basic hardware architecture of the timer section within the eTPU consists of either 16 or 32 channels.

All of the hardware is 24 bits wide. If a timer is just 16 bits wide you have to make a tradeoff between having enough resolution and enough range. Consequently, 16-bit timers need a prescaler, which lets you choose to have either a long range or have high resolution, but not necessarily both. 24-bit is the optimum width to cover most applications with high resolution and long range. 32-bit tends to be more than is typically required.

Each channel within the hardware timer includes two match registers, two comparators, two capture registers, and typically one I/O pin. Note that some implementations may include a second pin for feedback. This configuration allows the handling of very fast or very short pulses.

For example, for output, one match register can generate the first positive pulse, and the second register can generate the negative pulse. On input capture, the eTPU can capture the positive and then the negative edge. In fact, with some configurations it is possible to program the hardware to move data between the registers so that up to three high-speed events can be captured without needing servicing of the timer channel. These events are setup at the microcode level.

The timer channels work from one of two 24-bit counter references: TCR1 or TCR2. This retains compatibility with the original TPU, which also had the option of two reference counters. Although, now that the eTPU has 24 bits, it is not typically required to have them running at different speeds. We can also use one of the channels and one of the reference counters to generate an angular reference or angle clock which can be used for showing the angular position of an engine or the angular position of an AC motor.
Question

Which of the following statements about the eTPU are correct? Select all that apply and then click Done.

A. The eTPU is used for controlling different I/O functions.

B. The eTPU is not well-suited for advanced tasks, such as replacing logic in the I/O.

C. The eTPU is capable of address decoding if the logic or FPGA is directly connected to the bus.

D. The eTPU can be used for engine control.

Done

Take a moment to answer this question about the eTPU.

Correct.
The eTPU can be used for controlling different I/O functions. It can be used for advanced tasks, such as replacing logic in the I/O. While the eTPU is not capable of address decoding if the logic or FPGA is directly connected to the bus, it can be used for engine control.
Freescale provides a free library of eTPU functions including C Source Code, Host C API and detailed Application Note.

### eTPU Functional Library

<table>
<thead>
<tr>
<th>General Timing and Measurement</th>
<th>Communication</th>
<th>Motor Control</th>
<th>Automotive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pulse Width Modulation</td>
<td>Synchronous Peripheral Interface</td>
<td>Stepper Motor</td>
<td>Angle Pulse</td>
</tr>
<tr>
<td>Input Capture</td>
<td>UART</td>
<td>Motor Speed Control</td>
<td>Angle Clock</td>
</tr>
<tr>
<td>Output Compare</td>
<td>UART w/ Flow Control</td>
<td>DC Bus Break Control</td>
<td>Can Decode</td>
</tr>
<tr>
<td>Frequency and Pulse Measurement</td>
<td>Quadrature Decode</td>
<td>Fuel Control</td>
<td></td>
</tr>
<tr>
<td>Pulset Period Accumulator</td>
<td>Quadrature Decode</td>
<td>DC Speed Loop with QD</td>
<td>ACIM V/I/Hz Open Loop with SVM Drive</td>
</tr>
<tr>
<td>Synchronized PWM</td>
<td>Hall Sensor Decode</td>
<td>DC Speed Loop with HD</td>
<td>ACIM V/I/Hz Speed Loop with SVM Drive</td>
</tr>
<tr>
<td>Guarded Output Watch</td>
<td>Motor Control PWM</td>
<td>DC Speed and Current Loop</td>
<td>ACIM V/I/Hz Speed Loop with SVM Drive</td>
</tr>
<tr>
<td>General Purpose I/O</td>
<td>Current Control</td>
<td>BLCD with HD Open Loop</td>
<td>ACIM Vector Control with Speed Loop</td>
</tr>
<tr>
<td></td>
<td>Quadrature Decoder and Comparator</td>
<td>BLDC with HD Speed Loop</td>
<td>ACIM Vector Control with Speed Loop</td>
</tr>
<tr>
<td></td>
<td>Hall Sensor Decode Using Angle Mode</td>
<td>BLDC with HD Speed and Current Loop</td>
<td>PMBSM Torque Vector Control</td>
</tr>
<tr>
<td></td>
<td>ACIM V/I/Hz Control</td>
<td>BLDC with HD Speed Loop</td>
<td>PMBSM Vector Control with Speed Loop</td>
</tr>
<tr>
<td></td>
<td>ACIM Vector Control</td>
<td>BLDC with HD Speed Loop</td>
<td>PMBSM Vector Control with Speed Loop</td>
</tr>
<tr>
<td></td>
<td>PMBSM Vector Control</td>
<td>BLDC with HD Speed Loop</td>
<td>PMBSM Vector Control with Speed Loop</td>
</tr>
</tbody>
</table>

Freescale provides a free library of eTPU functions including C Source Code, Host C API and detailed Application Note.

### Electric Motors and Controls Supported

<table>
<thead>
<tr>
<th>DC Motors</th>
<th>AC Motors</th>
</tr>
</thead>
<tbody>
<tr>
<td>DC Motors</td>
<td>AC Motors</td>
</tr>
<tr>
<td>DC Motors</td>
<td>AC Motors</td>
</tr>
<tr>
<td>DC Motors</td>
<td>AC Motors</td>
</tr>
<tr>
<td>DC Motors</td>
<td>AC Motors</td>
</tr>
<tr>
<td>DC Motors</td>
<td>AC Motors</td>
</tr>
<tr>
<td>DC Motors</td>
<td>AC Motors</td>
</tr>
<tr>
<td>DC Motors</td>
<td>AC Motors</td>
</tr>
<tr>
<td>DC Motors</td>
<td>AC Motors</td>
</tr>
<tr>
<td>DC Motors</td>
<td>AC Motors</td>
</tr>
</tbody>
</table>

Web-based eTPU Function Selection Tool allows the generation of custom sets consisting any combination of available library functions within the limits of available eTPU memory resources.

Freescale provides a set of preprogrammed functions for the eTPU to make it easy for customers to use anything from simple to complex functions.

The General Timing and Measurement and Communications sets include functions from pulse width modulation to simple input capture and output compare, general purpose I/O, SPI, UART, and Synchronized PWM.

The Motor Control set is designed to give you the component functions that you need to control an electric motor. For example, there is a motor speed control, quadrature decode, hall sensor decode, motor control PWM, current control, and so on. By selecting the right combination of control functions, you can control virtually all available types of electric motors.

Stepper motor is a good example of the capabilities of the eTPU. With a conventional timer, the microcontroller has to service an interrupt at least once for every step of the stepper motor. It then has to do the calculations for controlling the output for the next step of the motor. With the eTPU, you simply initialize an acceleration profile for the stepper motor and then you program in the target position. For example, 350 steps to the right. The eTPU will step the motor through the acceleration, full speed, and deceleration phases to stop at the destination. If at any time you decide that you want a different destination position you can send that directly to the eTPU. The eTPU will then calculate what it must do to reach the new target position. Thus, instead of having to service the timer for every step of the motor you just send the destination without any additional servicing.

The Automotive set includes the various automotive functions. These functions can show the angle clock, decode the cam, control fuel, control sparks, and provide an angle reference pulse.

The second table lists all of the different motors. DC motors with brushes can be controlled open loop or closed loop. Brushless DC motors and AC motors can also be controlled. With AC motors we can take advantage of the eTPU’s capability to generate a DMA request and request sampling of the motor’s current by an A2D converter. This allows us to do closed loop AC motor control with no servicing from the host CPU.

On the current Coldfire 523X family this is limited to one motor because we only have one DMA request. We can control multiple motors in this fashion on other implementations where we can have multiple DMA requests from the eTPU. Even on the 523X we can control multiple motors, but we need the host CPU to provide the analog data for the control loop.

These library sets are available for download from the Freescale website at www.freescale.com/etpu.

Because we realize that the function sets that we provide are unlikely to fit all different application needs, there is also a web tool that allows a customer to configure a custom set of functions. This tool allows you to pick the combination of functions that you need and then generate a custom set specifically for the customer or for that application.
Future eTPU Functions

Freescale strives to expand the abilities of the eTPU by developing new functions to the eTPU Library

<table>
<thead>
<tr>
<th>Proposed Additions to eTPU Function Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>UART with Flow Control</td>
</tr>
<tr>
<td>I2S</td>
</tr>
<tr>
<td>HDLC</td>
</tr>
<tr>
<td>I2C</td>
</tr>
<tr>
<td>IrDA</td>
</tr>
<tr>
<td>USB Low-Speed Device</td>
</tr>
<tr>
<td>PCM DAC</td>
</tr>
</tbody>
</table>

We are also developing additional functions such as those listed here.

UART with flow control, which is based on the original version of the UART function. This is currently available.

The I2S function used for communicating to audio codecs. This is currently available.

Other functions that have been proposed for addition include HDLC, I2C, USB device low speed, PCM DAC, SIN/COS, Faster QDC, BLDC without commutation using valve drive; vector control, Hall Decoder with Angle Mode, BLDC Sensorless, Ramped PWM, and an eTPU Analyzer function.

Please visit our website to check on their availability or contact our marketing department for the exact status.
There are a lot of customers who have used the TPU on either 68300 family products or on PowerPC MPC500 family products. There have been two different masks of functions available. Mask A or Mask G. This table provides a cross-reference of equivalent functions from TPU Mask A to the new eTPU functions.

For example, the Angle-based engine control TPU function has a corresponding Fuel Control, Spark Control eTPU function.

Also, the High-time accumulation and Frequency divide/multiply TPU functions has a corresponding Pulse & Frequency Measurement eTPU function.

Lastly, the Digital input/output TPU function has a corresponding eTPU function of General Purpose I/O.

<table>
<thead>
<tr>
<th>TPU Function (Mask A)</th>
<th>Corresponding eTPU Function</th>
<th>Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Angle-based engine control</td>
<td>Fuel Control, Spark Control</td>
<td>2</td>
</tr>
<tr>
<td>Stepper motor control</td>
<td>Stepper Motor</td>
<td>1</td>
</tr>
<tr>
<td>Pulse-width modulation</td>
<td>Pulse Width Modulation</td>
<td>1</td>
</tr>
<tr>
<td>Frequency measurement</td>
<td>Pulse &amp; Frequency Measurement</td>
<td>1</td>
</tr>
<tr>
<td>High-time accumulation</td>
<td>Pulse &amp; Frequency Measurement</td>
<td>1</td>
</tr>
<tr>
<td>Frequency divide/multiply</td>
<td>Pulse &amp; Frequency Measurement</td>
<td>1</td>
</tr>
<tr>
<td>Pulse accumulator</td>
<td>Pulse Period Accumulate</td>
<td>1</td>
</tr>
<tr>
<td>Output compare</td>
<td>Output Compare</td>
<td>1</td>
</tr>
<tr>
<td>Input capture</td>
<td>Input Capture</td>
<td>1</td>
</tr>
<tr>
<td>Digital input/output</td>
<td>General Purpose I/O</td>
<td>1</td>
</tr>
</tbody>
</table>
### TPU Mask G/eTPU Functions Chart

<table>
<thead>
<tr>
<th>TPU Function (Mask G)</th>
<th>Corresponding eTPU Function</th>
<th>Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Queued Output Match</td>
<td>Queued Output Match</td>
<td>1</td>
</tr>
<tr>
<td>Fast Quadrature Decode</td>
<td>Quadrature Decode</td>
<td>3, 4</td>
</tr>
<tr>
<td>Multi-Phase Commutation</td>
<td>Quadrature Decoder &amp; Commutator</td>
<td>3</td>
</tr>
<tr>
<td>Input Capture/Transition Counter</td>
<td>Input Capture</td>
<td>1</td>
</tr>
<tr>
<td>UART</td>
<td>UART</td>
<td>1</td>
</tr>
<tr>
<td>Programmable Timer Accumulator</td>
<td>Pulse/Period Accumulate</td>
<td>1</td>
</tr>
<tr>
<td>Multi-Channel PWM</td>
<td>Synchronized PWM</td>
<td>1</td>
</tr>
<tr>
<td>Hall Effect Decode</td>
<td>Hall Sensor Decode</td>
<td>3, 4</td>
</tr>
<tr>
<td>Frequency Measurement</td>
<td>Pulse &amp; Frequency Measurement</td>
<td>1</td>
</tr>
<tr>
<td>Table Stepper Motor</td>
<td>Stepper Motor</td>
<td>1</td>
</tr>
</tbody>
</table>

Here are the equivalent eTPU functions for TPU Mask G. If you need to migrate from a device that used a TPU to an eTPU use this table to look up the equivalent function.

For example, the Queued Output Match TPU function has a corresponding Queued Output Match eTPU function.

Also, the Programmable Timer Accumulator TPU function has a corresponding Pulse /Period Accumulate eTPU function.

Lastly, the Table Stepper Motor TPU function has a corresponding eTPU function of Stepper Motor.
Here is an explanation of the functionality that is implemented within the eTPU and within the host. The host can be a 68K/Coldfire device or a PowerPC CPU. The eTPU has the same architecture. Let’s step through the structure, from the bottom up.

The target application hardware is connected directly to the timer channels of the eTPU.

The timer channels are serviced by the micro-engine. They request services through the scheduler. The scheduler schedules the micro-engine for the servicing.

The code for the servicing – which is typically written in standard C - is run out of the eTPU code memory. Although standard, this code has to be kept to a fairly strict code guideline, or code style, in order to verify that it works correctly with the very specific hardware on the eTPU.

The host interface includes all of the data parameters and is dual ported so that this data is accessible by both the eTPU and the host CPU.

On the host we have many APIs available that have been written in C. This allows users to easily interface with the eTPU function code running on the eTPU.

The classic API has been designed to approximate, as closely as possible, the host interface on the TPU. Customers who have software that interfaces to the TPU are recommended to use the classic API. Their software should need only minimal changes to be able to run on an eTPU based system.

For customers that don’t have legacy TPU code, we recommend using the full C API, which provides a very clean and high-level interface from their application.

The application would then be running on the host CPU, which again is either a 68K Coldfire or PowerPC.
Pulse Width Modulation (PWM)

The PWM eTPU function generates a Pulse Width Modulated waveform in which the frequency and duty cycle can be changed at any time by the CPU.

- Generate 0-100% duty cycle range
- Resolution of duty cycle is 0.01%
- Supports fixed and floating point duty cycle
- Select positive or negative polarity
- Immediate or next period update change of parameters
- Based on TPU PWM Function
- PWM API Functions include:
  - PWM Channel Initialization
  - PWM Duty Cycle Update
  - PWM Duty Cycle and Frequency Update
  - PWM Information

Example Usage for CPU:
```c
/* initialize channel PWM0 to: 10,000hz, 50%, active high, middle priority, using TCR1 */
fs_etpu_pwm_init( PWM0, FS_ETPU_PRIORITY_MIDDLE, 10000, 5000,
                 FS_ETPU_PWM_ACTIVE_HIGH, FS_ETPU_TCR1);
```

Sample C API:
```c
/* PWM channel initialization */
int8_t fs_etpu_pwm_init( uint8_t channel, uint8_t priority, uint32_t freq, uint16_t duty,
                         uint8_t polarity, uint8_t timer);
```

Here is a simple example of what the C API using PWM looks like. The API functions include the initialization of the channel, updating the duty cycle, updating both duty cycle and frequency, and providing PWM information.
Imagine you want to initialize Channel 0 on the eTPU to be PWM running at 10KHz 50% duty cycle active high, with high priority, and using Timer Counter Reference 1. Here is the exact API call and, within that, we can see that it says priority high.

First, we begin by selecting channel 0 and by setting the priority to high. Next, there’s 10,000 for 10KHz and 5,000 for 50% because we can select resolution band through a hundredth of a percent. And we show active high and eTPU Timer Counter Register 1.

This covers all of the required parameters in the initialization. There are similar interfaces for updating the duty cycle or the frequency. There are also similar interfaces for other functions.
eTPU Performance Example

- eTPU Performance with UART function:
  - If no other function is running on the eTPU
  - 830k baud with 37.5MHz eTPU
  - 1.45M baud with 66MHz eTPU
  - Each UART can run at any baud rate
  - A separate eTPU channel is required for each transmit and receive signal.

- Example
  - 37.5MHz eTPU (150MHz MCF523x)
    > 8 UARTs, 7 at 115k baud (tx and rx) and 1 at 19.2k baud (tx and rx) - (16 channels used)
    > 8 UARTs (tx and rx) at 103k baud (16 channels used)
  - 66MHz eTPU (132MHz MPC5500)
    > 16 UARTs, 12 at 115k baud (tx and rx), 3 at 19.2k and 1 at 9600 baud (32 channels used)
    > 8 UARTs (tx and rx) at 181k baud (16 channels used)

- Performance scales with frequency, maximum MCF523x CPU clock = 150MHz (37.5MHz eTPU clock)

One question that often arises is “what level of performance is available?” The type of function that loads the eTPU throughput the most tend to be serial interfaces. This is because we are really implementing a serial protocol using a timer. This requires the most servicing because we are programming the serial pulse generation through the timer.

If we were to program a UART with no other function running on the Coldfire 523X family, where the eTPU can run up to 37.5MHz with a single UART, we could work up to 830Kbaud. If we implement more than one UART, then the total throughput can still add up to 830Kbaud.

When running at 66MHz, which would be typical for the implementations on PowerPC and future Coldfire devices, we can go up to 1.45Mbit.

If there are several UARTs, they can run at different baud rates.

In this situation, a separate eTPU channel is required for each transmit and receive signal.

It is possible to have more transmitters than receivers, and vice-versa. In a typical example on the 523X family with 16 channels, we could implement eight UARTs, with seven running at 115Kbaud and one running at 19.2.

On a 66MHz version of the eTPU using 32 channels, we could implement 16 UARTs, with 12 running at 115Kbaud, three at 19.2, and one at 9.6.
Question

Is the following statement true or false? Click your answer and then click Done when you are finished.

“The eTPU functional library consists of functions for general and automotive purposes, and for DC and AC motors.”

True

False

Done

Consider this question regarding the eTPU functional library.

Correct.

Freescale provides a set of preprogrammed functions for the eTPU to make it easy for customers to use anything from simple to complex functions.
eTPU Function Selection Tool

Web-based tool to customize eTPU Function Selection

Allows for flexibility in eTPU configuration and usability

Supports all parts with eTPU (MCF523x and MPC55xx devices)

http://www.freescale.com/etpu

Although you may have the impression that the eTPU is complex to use, the truth is just the opposite. This is because Freescale provides easy-to-use libraries and web tools. One of the tools that makes this easy is the function selection tool. This tool allows you to easily pick your own combination of functions that you need for the application.

To use the tool, visit the eTPU web page at freescale.com/etpu and select the target device. Here we show the Coldfire MCF523X family. Use the menu to select any other device that includes an eTPU, and then click on all of the eTPU functions that you require. The tool will automatically check whether there is enough code and parameter memory on the target device to include all of the functions that you have selected.

You will then be prompted to provide some feedback about the applications and the use of these functions.

Next, click the compile button and the binary file of the custom set will be generated, together, along with some support files that enable this set to be used with the configuration tool.

Then you download these files to your system and move on to the next tool.

Because of an order from the United States International Trade Commission, BGA-packaged product lines and part numbers indicated here currently are not available from Freescale for import or sale in the United States prior to September 2010: MCF523X Products in 196 and 256 MAPBGA Packages.
The next tool is the eTPU graphical configuration tool. This tool provides a simple way of configuring the eTPU to perform the functions and functionality you may want straight from initialization. For example, a particular motor speed or a certain PWM value that you want to immediately appear on the output. Like the eTPU function selection tool, the graphical configuration tool is free to download.

You use the graphical interface to select the device you want to work on.

You select the eTPU function set, either one of the fixed function sets, or the custom function set that was generated using the web tool.

Then you set up the engine settings like Clock Sources and Prescalers for Time Base 1 & 2 and various Channel Input Digital Filter settings.

For each function you can assign functions to particular channels and select the configuration that you want.

Then you can configure the parameters for each respective eTPU function. For the UART function, you may select the priority, the baud rate, and the parity.
For customers or applications where the existing functions may not quite fit the requirements, it is possible to customize either one of the Freescale functions. To do this, we provide the source code and a detailed application note for each of the functions on our website.

Or, you can write your own function by either attending a training class on eTPU programming at Freescale or by buying the eTPU programming book from AMT Publishing.

There is also an eTPU compiler available from ByteCraft. This typically costs around $2,500.

There is also an eTPU simulator and debugger available from Ash Ware for around the same price.

The FreeMASTER development tool can also be used to support debugging and viewing signals.

All of the information and software and application notes are available on the specific eTPU web page available at www.freescale.com/etpu.
Question

Match each item to its description by dragging the letters on the left to their appropriate locations on the right. Click Done when you are finished.

A  FreeMASTER Application Development Tool  
Allows you to easily pick your own combination of functions that you need for an application.

B  eTPU Graphical Configuration Tool  
Provides a low cost virtual logic analyzer.

C  eTPU Function Selection Tool  
Provides a simple way of configuring the eTPU to perform the functions and functionality you may want straight from initialization.

Here is one more question about measures for minimizing problems caused by noise.

Correct.
The FreeMASTER Application Development Tool provides a low cost virtual logic analyzer. The eTPU Graphical Configuration Tool provides a simple way of configuring the eTPU to perform the functions and functionality you may want straight from initialization. The eTPU Function Selection Tool allows you to easily pick your own combination of functions that you need for an application.
Where Can I Learn More?

- eTPU Function Library and API
- eTPU Applications and Demonstrations
- eTPU Libraries Installation and Integration Guide
- eTPU Graphical Configuration Tool
- Links eTPU Compiler and Simulator Tools
- Link to eTPU VirtuaLab – Web interface to live demo
- Information on Trainings and Courses
- eTPU Documentation

Here is some more detailed information about what is available on the Freescale website. All of the documentation, including user manuals, and tools, such as the graphical configuration tool, are available from our website. The status of all of the functions is also available. Visit the site for all of the latest information.
Course Summary

- Basic understanding of the eTPU
- eTPU functions and architecture
- Functional libraries
- Graphical Configuration Tool
- Applications (motor control, connectivity, etc.)

**eTPU Tools are available on the Freescale website:**
http://www.freescale.com/etpu

This concludes this course on the Enhanced Time Processing Unit or eTPU. In this course you gained a basic understanding of the eTPU. You learned that the eTPU is very easy to program and that there is a wide variety of tools and resources available to ease eTPU use and development. You also learned about eTPU functionality and architecture, functional libraries, programming with the Graphical Configuration Tool, and eTPU applications such as motor control and connectivity.

Again, please visit the Freescale website for eTPU tools and for all of the latest information.