



Technical Note 39:
Scheduling the Protocol Translator

Technical Note 39: Scheduling the GPT

Technical Note 39: Scheduling the Protocol Translator	3
About the Protocol Translator	3
Activating the GPT	3
About User-supported Services	4
About GPT Processing Rules	4
Scheduling Considerations.....	4

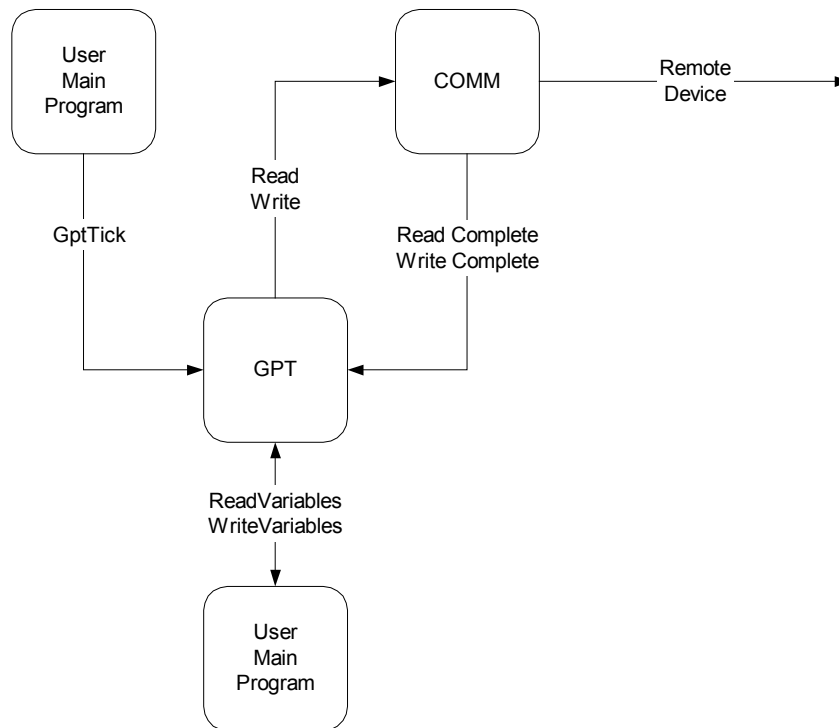
Technical Note 39: Scheduling the Protocol Translator

This technical note discusses protocol translator scheduling.

About the Protocol Translator

The ASE protocol translator consists of a generic framework (the generic protocol translator or GPT) and protocol-specific plug-in components, each of which supports a different protocol. For example, the DNPPT is the protocol-specific component that processes DNP 3.0 messages; the IECPT processes IEC messages, and so on. Each component can act as either an RTU or a master, depending upon the value of the property GPT_TRANSLATION. This document refers to the framework and the protocol-specific component together as the protocol translator or GPT.

The following figure shows the relationship between the user code and the GPT.



Activating the GPT

The user application activates the GPT using the following types of events:

- **Tick.** The user transmits a timer tick every 500 milliseconds. The GPT uses the timer tick to perform background processing, such as timing out a confirmed write.
- **ReadComplete.** The user application indicates to the GPT that a read that the GPT posted has completed successfully or unsuccessfully.
- **WriteComplete.** The user application indicates to the GPT that a write that the

Technical Note 39: Scheduling the GPT

GPT posted has completed successfully or unsuccessfully.

About User-supported Services

When the user invokes the GPT with an event, the GPT may request user-supported services. These services provide access to user process data and the physical communications port. The GPT uses these services only within the context one of the events described in “Activating the GPT” above. User supported services include the following:

- **COMM Read.** The GPT requests that the user read *n* bytes from the communications port. The user delivers the read completion event when the read completes.
- **COMM Write.** The GPT requests that the user writes *n* bytes to the communications port. The user delivers the write completion event when the write completes.
- **DAC Read/Write.** The GPT requests read or write access to one or more process variables to build a message for transmission to a remote device or to parse a message received from a remote device.

About GPT Processing Rules

The GPT follows these processing rules:

- **Non-blocking.** The GPT never blocks. The internal GPT code does not do any locking or waiting on event completions. Whenever the GPT needs to delay, it returns control to the user application. The user application delivers events when blocking activities complete.
- **CPU yielding.** The GPT never yields the CPU when processing an event. The system is almost always in user code, either waiting for an event or processing a service. When the user gives control to the GPT, the GPT does not yield the processor until the event has been processed. The GPT may invoke user application code while processing the event.
- **Context switching.** While the GPT is running, the user application is free to switch the context of the processor to a higher priority process. The user is responsible for locking to ensure that the switched process does not interfere with the state of the current request. For example, if the GPT is reading the user event queue and the user switches context to a higher priority process that inserts events into the queue, the user must maintain the integrity of the queue.

Scheduling Considerations

Consider the following when scheduling the GPT:

- **Running at high priority.** Some protocols require very high response time, such five to ten milliseconds. For these protocols, it is best that the GPT runs at a high priority. When processing an event, the GPT should run to completion. The context of the CPU should not change until the GPT has completely processed the event.
- **Running at normal priority.** For some protocols, such as DNP and IEC, timing

Technical Note 39: Scheduling the GPT

is not so critical. Responses can be delayed from 20 to 50 milliseconds. For these protocols, the user application can employ one of two types of scheduling, depending on the constraints of target OS:

- **Pre-emptive scheduling.** In this scenario, higher priority activities can interrupt the GPT. Control returns to the GPT when the higher priority tasks complete.
- **Yielding.** The user application yields whenever the GPT enters the user code. That is, whenever the GPT makes a COMM or DAC request, the user yields the processor to higher priority tasks. The user application must test to ensure that a maximum amount of time is not exceeded between yields. Note that there is no code within the GPT that enforces a maximum amount of time before yielding the CPU.