Service Based Bypass is the bypass method best used when the bypass needs to exactly define at which point in the ECU's data flow data shall be sampled and/or a bypass shall be triggered fully synchronously to a ECU function and raster delay shall be avoided.
As a consequence, each bypassed function requires its own ECU trigger, which can be a limiting factor in some ECU projects. This paper describes how the available ETK resources can be used in an efficient way.

The background

When data has been copied from the ECU into the ETK managed memory, an ETK trigger is issued to inform the ES910 that new input data from this service point is available. This ETK trigger raises an event in the ES910 that can be used to copy the data from the ETK buffer to the ES910 memory and to trigger an ES910 bypass process. Thus, it synchronizes ECU scheduling with the ES910.
For writing back data to the ECU, there's no such trigger as the ECU's real time behavior shall not be changed. Instead, data is written back to the ETK managed bypass output buffer. It's up to the ECU to check if new data is available whenever it's appropriate. If a timeout has been defined in the configuration of the service point, the ECU will check for new bypass data till that timeout has been reached.

A simple example

Let's look at a simple example to explain how it works. In this setup, there are three service points with different configurations: At the first service point, data is read and written back, the second service point only reads data from the ECU and the third service point only writes back data to the ECU. Whether the read / write accesses are defined before or after the service point does not matter for this example. Here's how the configuration looks like in INTECRIO or ASCET-RP's HWC:

	Name	Priority			Timeout [ms]	Tolerated Lost Cycles	Reaction On Lost Communi				Timeout [ms]	Tolerated Lost Cycles	Reaction On Lost Communi
1	ProcB_ServicePoint	1	☑	☐				☑	☐	☑	0.5	1	Execute
2	ProcE_ServicePoint	2	☑	☐				☑	☐	☐			
3	ProcG_ServicePoint	3	☐	☐				☑	☐	☑	0.5	1	Execute

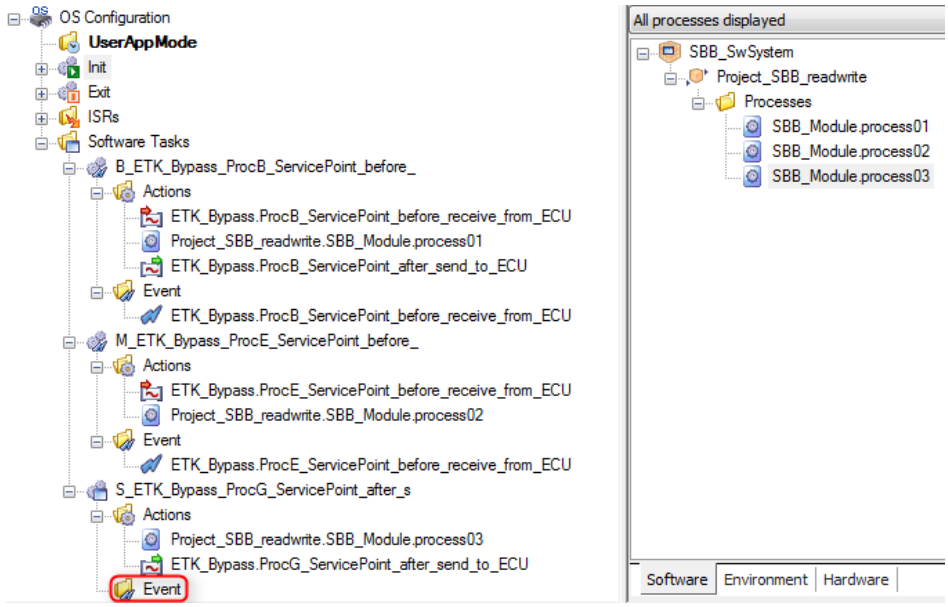
Parameters | Service Point Selection | Hooked Service Point Selection | Signal Selection | Service Points | Signal Groups | Signals

Three different service point configurations

As we can see, setting the time out is only activated for the "write back to ECU" actions. In fact, this setting actually calibrates the ECU service point implementation and can also be changed by INCA at runtime of the bypass experiment.

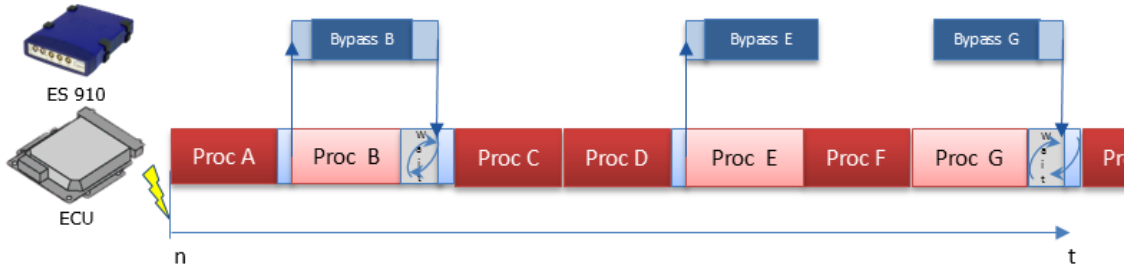
Each configured "read from ECU" action will result in a trigger event in the ES910 RTA-OSEK real time operating system. However, no timing on the ES910 is defined at this point!

The OS events created by this set up can be seen in the „OS Configuration" view in INTECRIO. Since the third service point only defined a „write back to ECU, there's no event created.



Events and actions created by the configuration in the OS configuration view of INTECRIO

The read and write processes created by the "read from ECU" and "write back to ECU" hooks in the "service point selection and configuration" tab are assigned to the available software tasks when the "OS auto config" feature is used. However, for the ES910 OS, they are like any other processes and can be assigned to any other task. Likewise, the bypass software processes can be assigned to any (usually the created event triggered) task. This will result in a timing as shown here:

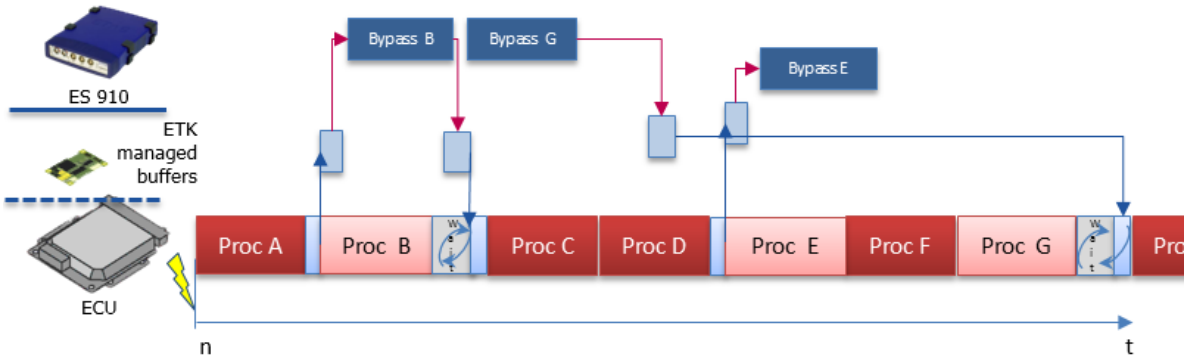


Triggering pf external bypass processes and ECU timing

Note that there is no event for the third service point, since no "read from ECU" has been configured and therefore there is no trigger event created. This means that none of the action assigned to this task will be executed, as the task will never be triggered!
It either needs to be defined when the task shall be executed, or the processes in this task need to be assigned to another task.

Triggering of the task can either be achieved by using the RTA-OSEK "ActivateTask()" command or by configuring a dummy "read from ECU" at the service point, which then would give us a trigger event. However, if this is done for several service points, it will cause unnecessary interrupt load to the ES910's OS. Since the set up only intends to define when bypass outputs are written to the ECU, separating the bypass calculation from the "write back to ECU" process allows the ES910 to calculate the bypass in another task with a lower priority.

This way, the ES910 is free to schedule the execution a some other time, making better use of its resources without any impact to the intended behavior. This allows a better load balancing in a system set up with many service points and a high bypass calculation load, optimizing the use of ES910's performance.
Here is an example of the communication, if the bypass process of Proc_G would be added to the same event as Proc_B ("read fro ECU at service point B") – avoiding another interrupt event caused by a dummy "read from ECU":



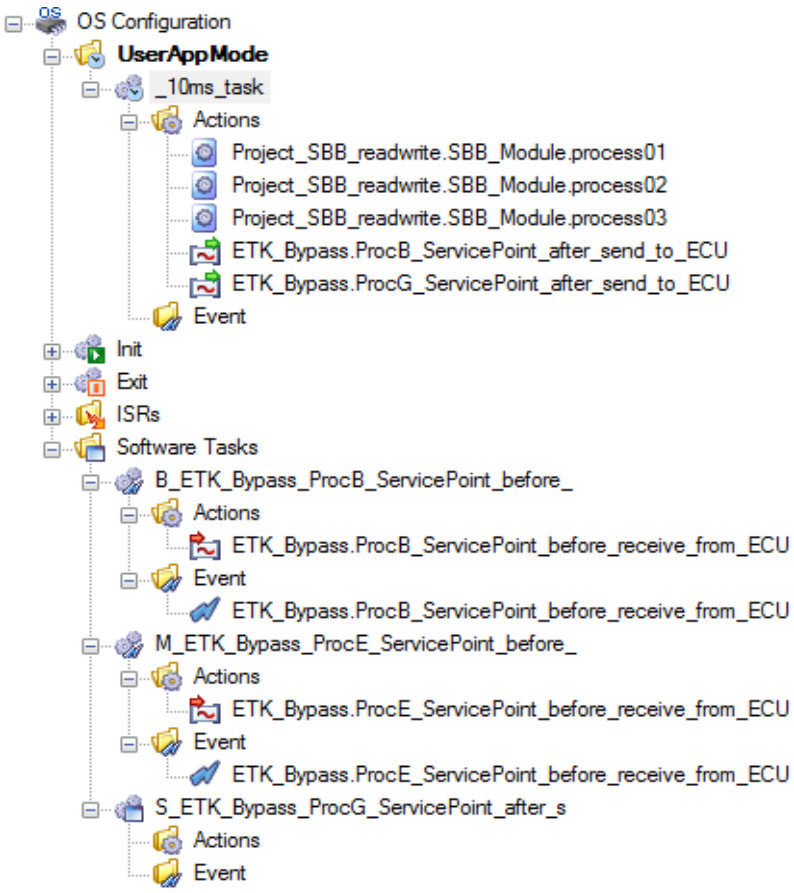
Triggering of external bypass processes and ECU timing

If the bypass use case requires reading from and/or writing data to the ECU at clearly defined points in the ECU's data flow, but no synchronized bypass trigger, the calculation can be completely decoupled from the ECU's scheduling and communication, like in the example on the right.

The bypass is calculated in a time triggered task, while the communication still takes place at clearly defined points. The "Write back" processes are called in the 10ms timer task of the ES910.

The set up minimizes task management overhead and allows optimal use of the ES910's calculation power. The time outs in the ECU can be set to 0 to minimize effects on ECU timing behavior also.

Note: The examples have been set up using INTECRIO, however, the same configurations can be done in ASCET-RP using the OS-Editor.



Fully decoupled bypass calculation

Tools used

- ASCET-RP
- INTECRIO
- INTECRIO-RLINK
- ES910

Your ETAS Contact

Markus Gebhardt; +49 711 3423 - 2278
Markus.gebhardt@etas.com