

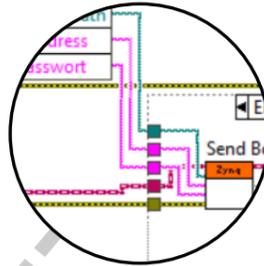
JavaScript

```

response=await
response.ok){
return false;
}
info=await info_response.json;
reg=null;
(navigator.serviceWorker){
reg=await navigator.service
sw=reg?reg.active:null;
res=new Promise(resolve=>{
let ws=new WebSocket(info
ws.onopen=(ev)=>{
let sdsp=new S
resolve(sdsp)
}
}
}
error=(ev)=>{

```

LabView



Python

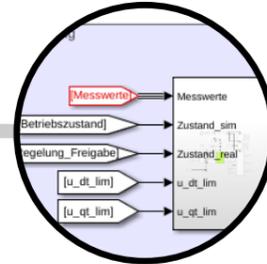
```

dtype=int
ch.Maxv['max']
ch.Minv['min']
ch.MaxIncrementv['max']
self.varinfo.append(ch)
if len(self.varinfo) == 0: #
return
dt_list=list((x.Name, '<i4' if x
dt=np.dtype(dt_list)

self.callback_info(self.sampled
data con rx, data con tx=await
await asyncio.sleep(1) # Wait
self.running=True
while self.running:
try:
data=await data

```

Simulink



Rust

```

fn info(
dsp: web::Data<Add>,
app_info: web::Data<
req: HttpRequest,
) -> Result<impl Responder,
let info = dsp
.send(dsp_manager::
.await
.map_err(|_| actix
if let dsp_manager::b
let dsp_info =
.map_err(|
let response

```

Embedded C++

```

const boot::File6 file
begin def:erstan Section
uint32_t* sectionStart = reinterpret_cast<co
uint32_t length = sectionStart[1]; // Sect
uint32_t* begin = sectionStart + 2; //
std::uint32_t* end = sectionStart + length + 2; //
uint32_t entryPoint = address; // erste Section sollte de
address |= 0xFFFFFFFF && length |= 0xFFFFFFFF) {
d::copy(begin, end, reinterpret_cast<std::uint32_t*>(
0x00000000), address, length = sizeof<std::uint32_t
const Section : *
"0x" << std::setw(8) << std::hex << std::setw(8) <<
std::setw(8) << std::dec << std::setfill(
<< length + sizeof<std::uint32_t>
);
};

```

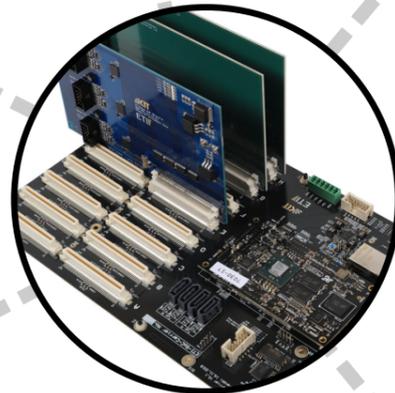
VHDL

```

-- M_AXIS_VALID ist solange '1' folgende Komponente Daten
p_a_last : process(Axis_ACLX)
begin
if rising_edge(Axis_ACLX) then
if Axis_Ansden = '0' then
m_axis_last <= '1';
end if;
else
m_axis_ready <= '1';
end if;
end process p_a_ready;

-- M_AXIS_VALID ist solange '1' folgende Komponente Daten
p_a_last : process(Axis_ACLX)
begin
if rising_edge(Axis_ACLX) then
if Axis_Ansden = '0' then
m_axis_last <= '1';
end if;
else
m_axis_ready <= '1';
end if;
end process p_a_last;

```



Das ETI hat ein Rapid-Prototyping-System auf Basis eines Zynq-SoC von Xilinx entwickelt. Dieses SoC beinhaltet einen FPGA und zwei ARM-A9 Kerne. Das System ist über Ethernet mit einem Steuerrechner verbunden um Sollwerte zu setzen und Werte zu visualisieren.

All dies benötigt diverse Software auf allen Abstraktionsebenen, die ständig weiterentwickelt werden kann. Hierfür bieten wir ein Forschungspraktikum in Einzel- oder Gruppenarbeit an.

Bei Interesse wenden Sie sich bitte an eine dieser Personen:

Philip Kiehnle (Raum 103, philip.kiehnle@kit.edu)

Patrick Himmelmann (Raum 103, pat.him@kit.edu)

Stefan Mersche (Raum 118, stefan.mersche@kit.edu)

Benedikt Schmitz-Rode (Raum 015, schmitz-rode@kit.edu)