

# On Extending IPXACT for Device Driver Software Generation

Sandeep Pendharkar,  
sandeep@vayavyalabs.com

## 1. Introduction

IPXACT[2] is a specification by the SPIRIT[1] consortium for describing an IP. The IPXACT standard describes an XML data format and structure documented with a schema for capturing meta-data that describes an IP used in development, implementation and verification of electronic systems. It provides a consistent mechanism to capture all aspects of an IP thus facilitating automatic configuration, integration and packaging of IPs by automation tool.

There have been various attempts to extend the scope of IPXACT beyond its current form for various ESL activities. At Vayavya Labs, we are working on extending the domain of ESL beyond the hardware to automatically synthesize the associated software as well. Our product DDGEN automatically synthesizes the device driver software for a given peripheral from an input specification called DPS (Device Programming Specification).

We believe the scope of IPXACT should be extended to capture this device specification which is currently specified in DPS. Towards this goal, we have also designed our Vendor Extensions to the current IPXACT schema.

This document is our expression of intent to discuss our Vendor Extensions in appropriate forums/initiatives with the intention of getting feedback and improving based on this feedback. Vayavya Labs is committed to extending IPXACT beyond its current scope into the domain of software synthesis and wishes to participate in similar projects and initiatives.

Section 2 provides a brief introduction to DDGEN while section 3 provides an illustrative example of our Vendor Extensions.

## 2. About DDGEN

Device driver development is often error-prone, complex, time-consuming and often a bottleneck in embedded system projects due various reasons - strict and aggressive schedules, plethora of operating systems and driver models to name a few. DDGEN relieves the device driver writer of this tedium by automatically generating the device driver.

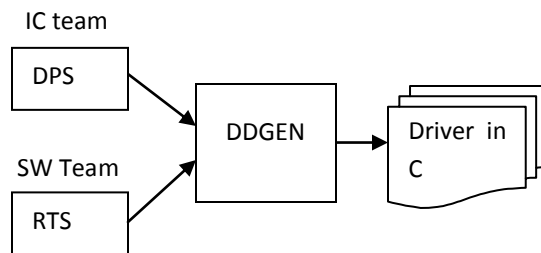


Fig (1)

As shown in Fig(1), DDGEN requires two input specifications viz. DPS and RTS. DPS(Device Programming Specification) captures all relevant aspects(registers, interrupts, fifos, programming sequence etc.) of a given device. Thus DPS is a formal specification language. RTS captures software considerations like the target operating system, driver model, interrupt handling mechanism etc. DDGEN compiles both the input specifications and automatically synthesizes the device drivers. Refer [3] for a complete discussion on DDGEN.

### 3. DPS specific Vendor Extensions to IPXACT

In its current form, the IPXACT specification does not allow capturing of the device specific details required from driver generation point of view. Except for the register specification, there is nothing common between IPXACT and DPS. As such, we have designed Vendor Extensions for every specification in DPS except the register specification. Note that in a given IPXACT xml, data captured by these Vendor Extensions will co-exist along with the other data related to that particular IP. Given below is an illustrative example of an IPXACT Vendor Extension for the Interrupt Specification in DPS. Fig (2) shows the syntax for the Interrupt Specification in DPS in Extend BNF. The corresponding IPXACT Vendor Extension is show in Fig (3). Refer [4] for the complete schema of our Vendor Extensions.

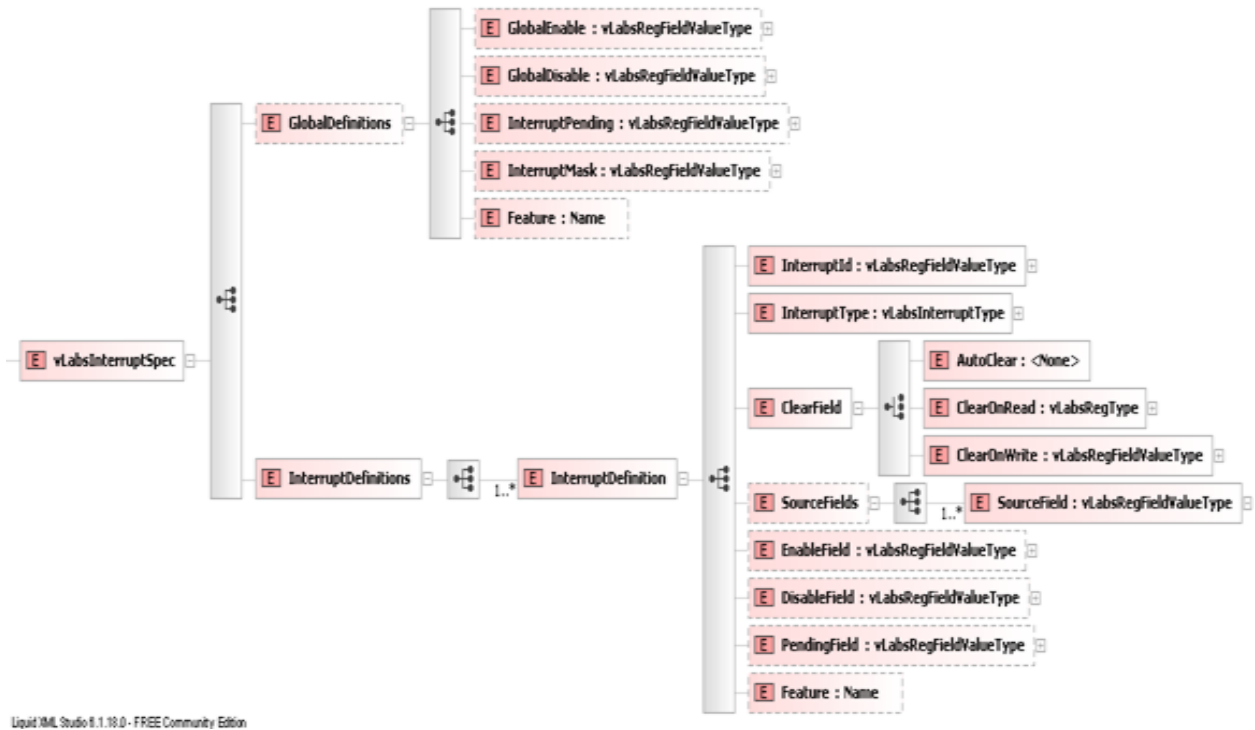
```
dps_interrupt_specification:=
  interrupt_spec
  {global_interrupt_fields interrupt_definitions}
}

global_interrupt_fields:=
  [global_enable = reg_field(bit_field);]
  [interrupt_pending = reg_field (bit_field);]
  [interrupt_mask = reg_field;]
  [feature_name = identifier;]

interrupt_definitions := interrupt_definition {interrupt_definition}

interrupt_definition :=
  interrupt_id {
  int_type = device_read | device_write | status | error ;
  clear_field = auto_clear | cor reg_name | cow reg_field(bit_field) ;
  [source_field = reg_field(bit_field) {|| reg_field(bit_field)} ;]
  [enable_field = reg_field(bit_field) ;]
  [disable_field = reg_field(bit_field) ;]
  [pending_field = reg_field(bit_field) ;]
  [feature_name = identifier ;] }
```

Fig(2)



Fig(3)

The Interrupt Specification in DPS captures all details about different types of interrupts, mechanism to enable/disable/clear them. DDGEN uses this information in synthesizing the ISR (Interrupt Service Routine) part of the driver code. Refer [5] for a detailed explanation of the Interrupt Specification in DPS

#### 4. Conclusion

IPXACT in its current form captures IP details meant primarily for IP design, verification, packaging and integration only. Extending IPXACT to capture details for software synthesis is a logical evolution. We have designed a set of Vendor Extensions to IPXACT such that DDEN automatically generates device driver code from these Vendor Extensions. Vayavya Labs is keen to share these Vendor Extensions with a wider community and improve them based on the feedback provided by this community. We hope that our Vendor Extensions eventually get incorporated in the IPXACT standard.

#### References:

- [1] Spirit Consortium - <http://www.spiritconsortium.org/home/>
- [2] IPXACT 1.5 Draft.
- [3] DDGEN: An Automated Device Driver Generation Tool for Embedded Systems, IP-ESC 09, Grenoble, France.
- [4] IPXACT Vendor Extensions for DDGEN:  
[http://www.vayavyalabs.com/download\\_sample/fdwn.php?file=VLabsSchema.xsd](http://www.vayavyalabs.com/download_sample/fdwn.php?file=VLabsSchema.xsd)
- [5] DPS Language Reference Manual version 1.0