Zephyr® Project
Developer Summit 2022

June 8-9, 2022
Mountain View, CA + Virtual

USB-C Power Delivery Sink Device

Creating a USB Type-C Sink Device with Zephyr

Sam Hurst (shurst@google.com)

Contents

Zephyr® Project
Developer Summit 2022

June 8-9, 2022
Mountain View, CA + Virtual

Power Source Solutions

Antiquated Power Source Solutions

Barrel Jacks

USB-B and variants

Advantages:

- Simple

Disadvantages:

- Bulky
- Fixed Power
- Barrel Jack non-standard sizes
- Outdated

Zephyr™

USB Type-C Power Source Solutions

Advantages:

- Small profile
- Variable Power
- Symmetrical
- Alternate Modes

Disadvantages:

- Complex

Zephyr™

USB Type-C Power Source Solutions

Advantages:

- Small profile
- Variable Power
- Symmetrical
- Alternate Modes

Disadvantages:

- Complex(**Mitigated by the Zephyr USB-C Subsystem**)

Zephyr™

Zephyr® Project
Developer Summit 2022

June 8-9, 2022
Mountain View, CA + Virtual

USB-C Jargon

Port Data Role

- A port can be a Downstream Facing Port (DFP) or an Upstream Facing Port (UFP)
- A DFP is viewed as the USB Host
- A UFP is viewed as the USB Device
- The Port Data Role is tracked and maintained in software
- The Port Data Roles can switch

Port Power Role

- A port can be a Source or a Sink
- Source port supplies power
- Sink port consumes power
- The Port Power Roles can switch
- Power is supplied over VBUS

Source Capabilities and Sink Request

- A Source sends its Capabilities to a Sink in the form of Power Data Objects (PDOs)
- A Sink requests one of the PDOs by sending a Request Data Object (RDO) to the Source

Zephyr® Project
Developer Summit 2022

June 8-9, 2022
Mountain View, CA + Virtual

USB-C Type-C Sink

On initial connection between Source and Sink before PD

- Depending on the Source, it provides the Sink one of the following Non-PD power levels based its Rp:
  - 5V@3A
  - 5V@1.5A
  - 5V@0.5A

- At this point, if these power levels are sufficient, nothing else needs to be done

Zephyr® Project
Developer Summit 2022

June 8-9, 2022
Mountain View, CA + Virtual

USB-C Power Delivery Sink

After initial connection, Source sends Capabilities to Sink

- Source Capabilities are represented by a sequence of PDOs
- Four types of PDO: Fixed, Variable, Battery, and Augmented
- Given the type, different capabilities are available to the Sink

Fixed Supply PDO

- A well-regulated fixed voltage power supply
- Encoded as a 32-bit value
- Most common type
- Typical Fixed Supply PDOs for a 45W charger:
  - PDO1: 5V@3A
  - PDO2: 9V@3A
  - PDO3: 15V@3A
  - PDO4: 20V@2.25A

NOTE: PDO1 is always 5V

Bit(s)	Description
B31...30	Fixed supply
B29	Dual-Role Power
B28	* USB Suspend Supported
B27	Unconstrained Power
B26	* USB Communications Capable
B25	Dual-Role Data
B24	* Unchunked Extended Messages Supported
B23...22	Reserved – Shall be set to zero.
B21...20	Peak Current
B19...10	Voltage in 50mV units
B9...0	Maximum Current in 10mA units

* out of scope for the presentation

Sink Request Data Object (RDO)

- Four types of RDO: Fixed, Variable, Battery, and Augmented
- Depending on the type of PDO selected, a matching RDO of the sent type must be used

Fixed RDO

- Used when a Fixed PDO is selected
- Encoded as a 32-bit value

Bit(s)	Description
B31	Reserved – Shall be set to zero
B30...28	Object position (000b is Reserved and Shall Not be used)
B27	* GiveBack flag = 0
B26	Capability Mismatch
B25	* USB Communications Capable
B24	* No USB Suspend
B23	* Unchunked Extended Messages Supported
B22...20	Reserved - Shall be set to zero.
B19...10	Operating current in 10mA units
B9...0	Maximum Operating Current 10mA units

* out of scope for the presentation

Zephyr® Project
Developer Summit 2022

June 8-9, 2022
Mountain View, CA + Virtual

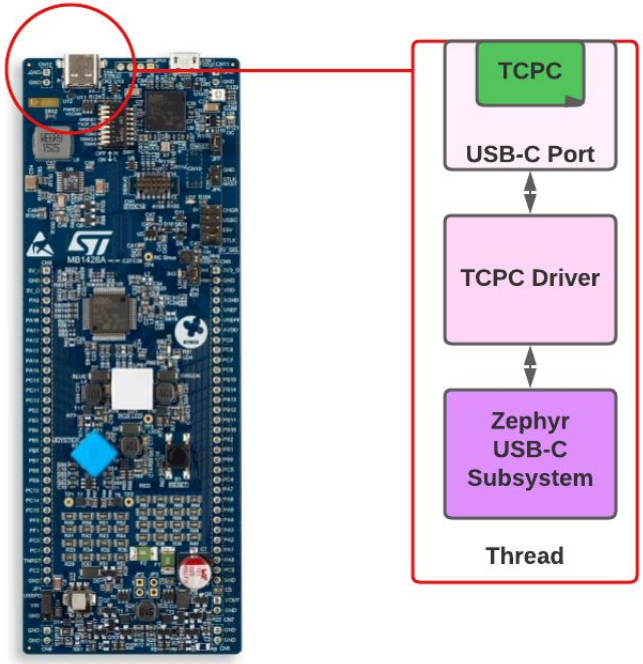
USB-C Support in Zephyr

Port of Google's Chrome OS USB-C Stack

- Initial Chrome OS USB-C stack written in 2014
  - Advantages
    - Small footprint
  - Disadvantages
    - Monolithic design
    - Non-standard naming convention of features

- Total rewrite of the Chrome OS USB-C stack was done in 2019
  - Advantages
    - Modular design
    - Closely follows the specification for naming features
  - Disadvantages
    - Larger footprint

Logical View of a USB-C Port



Devicetree USB-C Port Representation

- Each physical Type-C Port is described by a Devicetree node that's compatible with **"usbc-port"**
- Each Type-C Port has its own thread

Example Devicetree node:

```
port1: usbc-port@1 {
      compatible = "usbc-port";
      label = "PORT1";
      reg = <1>;
      /* Type-C Port Controller: STM32 UCPD1 */
      tcpc = <&ucpd1>;
};
```

Minimum USB-C API

- `void usbc_set_policy_cb_set_src_cap(const struct device *dev, policy_cb_set_src_cap_t policy_cb_set_src_cap)`
- `void usbc_set_policy_cb_get_request_data_object(const struct device *dev, policy_cb_get_request_data_object_t policy_cb_get_request_data_object)`
- `void usbc_set_policy_cb_check(const struct device *dev, policy_cb_check_t policy_cb_check)`
- `void usbc_set_policy_cb_notify(const struct device *dev, policy_cb_notify_t policy_cb_notify)`
- `int usbc_start(const struct device *dev)`

See include/zephyr/usbc/usbc.h for entire API

Zephyr® Project
Developer Summit 2022

June 8-9, 2022
Mountain View, CA + Virtual

USB-C Zephyr Sink Application

Requirements

- Policy
  - Sink only
  - PD Power 5V@100mA
  - UFP only

- Application
  - Display Source Capabilities on the serial console

Source Capabilities Callback

```
static void usbc_port1_policy_cb_set_src_cap(const struct device *dev, uint32_t *pdos, int num_pdos)
{
	int i;


	if (num_pdos > PDO_MAX_DATA_OBJECTS) {
		num_pdos = PDO_MAX_DATA_OBJECTS;
	}


	for (i = 0; i < num_pdos; i++) {
		src_caps[i] = *(pdos + i);
	}

	src_cap_cnt = num_pdos;
}
```

Request Data Object Callback

```
static uint32_t usbc_port1_policy_cb_get_request_data_object(const struct device *dev)
{
	union pd_rdo rdo;

	rdo.fixed.min_or_max_operating_current = PD_CONVERT_MA_TO_FIXED_PDO_CURRENT(100);

	rdo.fixed.operating_current = PD_CONVERT_MA_TO_FIXED_PDO_CURRENT(100);

	rdo.fixed.unchunked_ext_msg_supported = 0;
	rdo.fixed.no_usb_suspend = 1;
	rdo.fixed.usb_comm_capable = 0;
	rdo.fixed.cap_mismatch = 0;
	rdo.fixed.giveback = 0;

	rdo.fixed.object_pos = 1;

	return rdo;
}
```

Simply selects PDO1 (5V).
Typical app. would analyze the Source Capabilities and select a PDO

Policy Check Callback

```
bool usbc_port1_policy_check(const struct device *dev, enum policy_check_t policy_check)
{
	switch (policy_check) {
		case CHECK_POWER_ROLE_SWAP:
			return false;

		case CHECK_DATA_ROLE_SWAP_TO_DFP:
			return false;

		case CHECK_DATA_ROLE_SWAP_TO_UFP:
			return false;

		case CHECK_SNK_AT_DEFAULT_LEVEL:
			return true;

		default:
			return false;
	}
}
```

Policy Notify Callback

```
static void usbc_port1_notify(const struct device *dev, enum policy_notify_t policy_notify)
{
      switch (policy_notify) {
            ...
            case TRANSITION_PS:
                  display_source_caps(dev);
                  break;
            ...
      }
}

Other cases:

      PROTOCOL_ERROR,              MSG_DISCARDED,     MSG_ACCEPT_RECEIVED,    MSG_REJECTED_RECEIVED,
      MSG_NOT_SUPPORTED_RECEIVED,  PD_CONNECTED,      NOT_PD_CONNECTED,       POWER_CHANGE_0A0,
      PORT_PARTNER_NOT_RESPONSIVE, POWER_CHANGE_DEF,  POWER_CHANGE_1A5,       POWER_CHANGE_3A0,
      SNK_TRANSITION_TO_DEFAULT,   DATA_ROLE_IS_UFP,  DATA_ROLE_IS_DFP,       HARD_RESET_RECEIVED
```

Board Configuration and Port Device

```
/* Config the board */
ret = board_config();
if (ret) {
      LOG_ERR("Could not configure board");
      return;
}


/* Get the device for this port */
usbc_port1 = DEVICE_DT_GET(DT_NODELABEL(port1));
if (!device_is_ready(usbc_port1)) {
      LOG_ERR("PORT1 device not ready\n");
      return;
}
```

Register the Callbacks

```
/* Register Policy Check callback */
usbc_set_policy_cb_check(usbc_port1, usbc_port1_policy_check);


/* Register Policy Notify callback */
usbc_set_policy_cb_notify(usbc_port1, usbc_port1_notify);


/* Register Policy Set Source Capabilities callback */
usbc_set_policy_cb_set_src_cap(usbc_port1, usbc_port1_policy_cb_set_src_cap);


/* Register Policy Get Request Data Object callback */
usbc_set_policy_cb_get_request_data_object(usbc_port1, usbc_port1_policy_cb_get_request_data_object);
```

Start the USB-C Subsystem

```
/* Start the USB-C Subsystem for Port1 */
usbc_start(usbc_port1);

while (1) {
      /* Perform Application Specific functions */

      /* Arbitrary delay */
      k_msleep(5000);
}
```

Calling usbc_start actual starts the thread used to run the USB-C subsystem on for the given port.

Application Demo Output

USB-C Charger

STM32 B_G747E_DPOW1

```
PDO 1:
        Type:               FIXED
        Current:            3000
        Voltage:            5000
        Peak Current:       0
        Unchunked Support:  0
        Dual Role Data:     0
        USB Comms:          0
        Unconstrained Pwr:  1
        USB Suspend:        0
        Dual Role Power:    0
PDO 2:
        Type:               FIXED
        Current:            3000
        Voltage:            9000
        Peak Current:       0
        Unchunked Support:  0
        USB Comms:          0
        Unconstrained Pwr:  0
        USB Suspend:        0
PDO 3:
        Type:               FIXED
        Current:            3000
        Voltage:            15000
        Peak Current:       0
        Unchunked Support:  0
        USB Comms:          0
        Unconstrained Pwr:  0
        USB Suspend:        0
PDO 4:
        Type:               FIXED
        Current:            2250
        Voltage:            20000
        Peak Current:       0
        Unchunked Support:  0
        USB Comms:          0
        Unconstrained Pwr:  0
        USB Suspend:        0
```

Zephyr™

Status and Todo

- Status of Zephyr USB-C Subsystem
  - PR under review: https://github.com/zephyrproject-rtos/zephyr/pull/45601
  - Should land in Zephyr Version 3.2

- Todo
  - Implement additional features: Source, Dual-Role Power, Alt-Modes, GiveBack, etc.
  - Move policy decisions from code to Devicetree
  - Add unit tests

Additional Resources

- USB Type-C Specification
- USB Power Delivery Specification
- USB Type-C Port Controller Interface Specification
- Pull Request

Zephyr® Project
Developer Summit 2022

June 8-9, 2022
Mountain View, CA + Virtual

Questions?