SAMSUNG

Proposal for I3C SW Support in Zephyr RTOS

Shashank Prashar, YeEun Kim

Samsung Semiconductor India R&D Center

Agenda

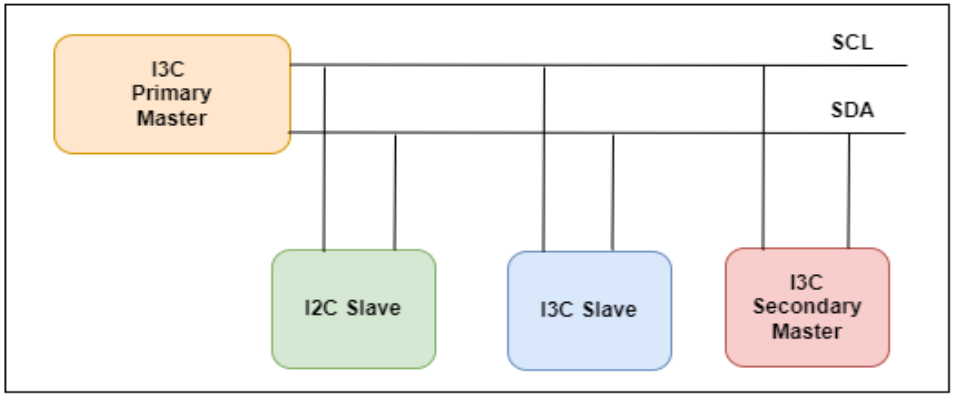
Zephyr® Project

- Overview of I3C
  - The I3C Protocol
  - I3C Features
- Why I3C over Current SW support in Zephyr ?
  - A Typical I2C-SPI system vs I3C System
- Proposal for I3C in Zephyr
  - SW File Structure
  - Master Driver Probing Sequence
  - Master Write - Read Sequence

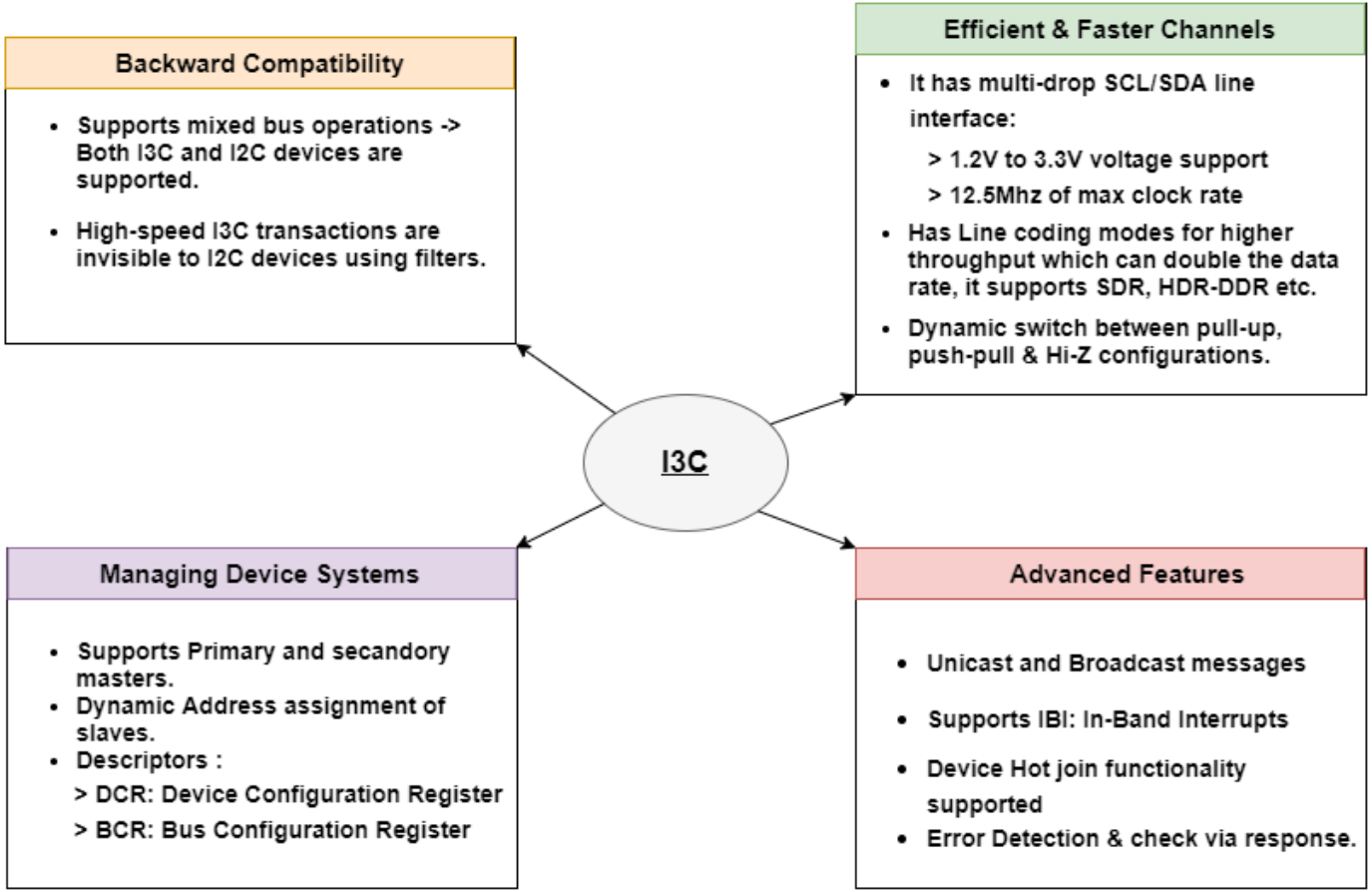
I3C : Protocol

I3C = I + I2C (The improved I2C)

- The SCL clock line is in push-pull configuration
- SDA data line in open drain with pull up
- SDA still can switch to push-pull configuration for high throughput
- Max clock rate of up to 12.5 MHz
- Availability of In Band
- Supports multiple classes of devices
  - Main master
  - Secondary master
  - I3C slave
  - I2C slave

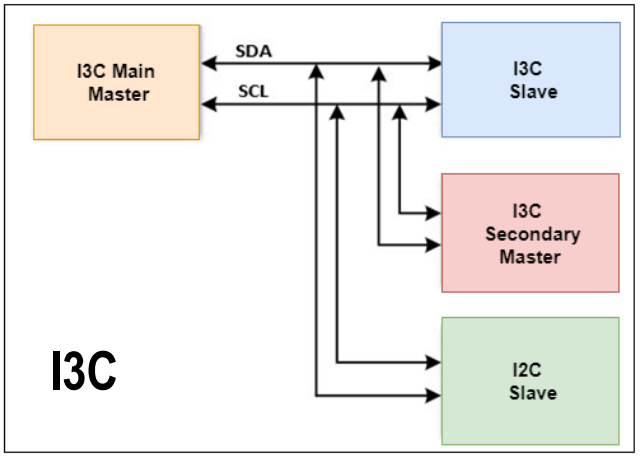


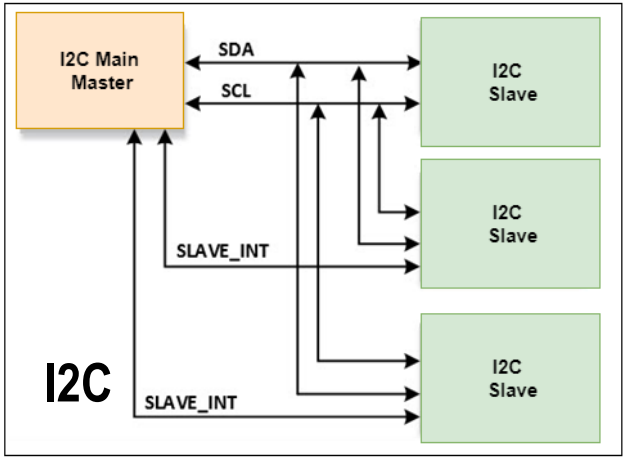
I3C Features

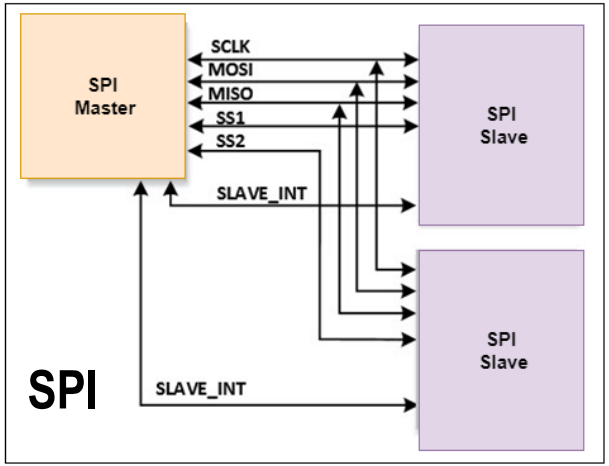


Comparison – I3C, I2C and SPI

- I3C – Only 2 wires are required
- I2C – 2 wires and separate wire for each interrupt signal
- SPI – 4 wires and separate wire for each interrupt signal



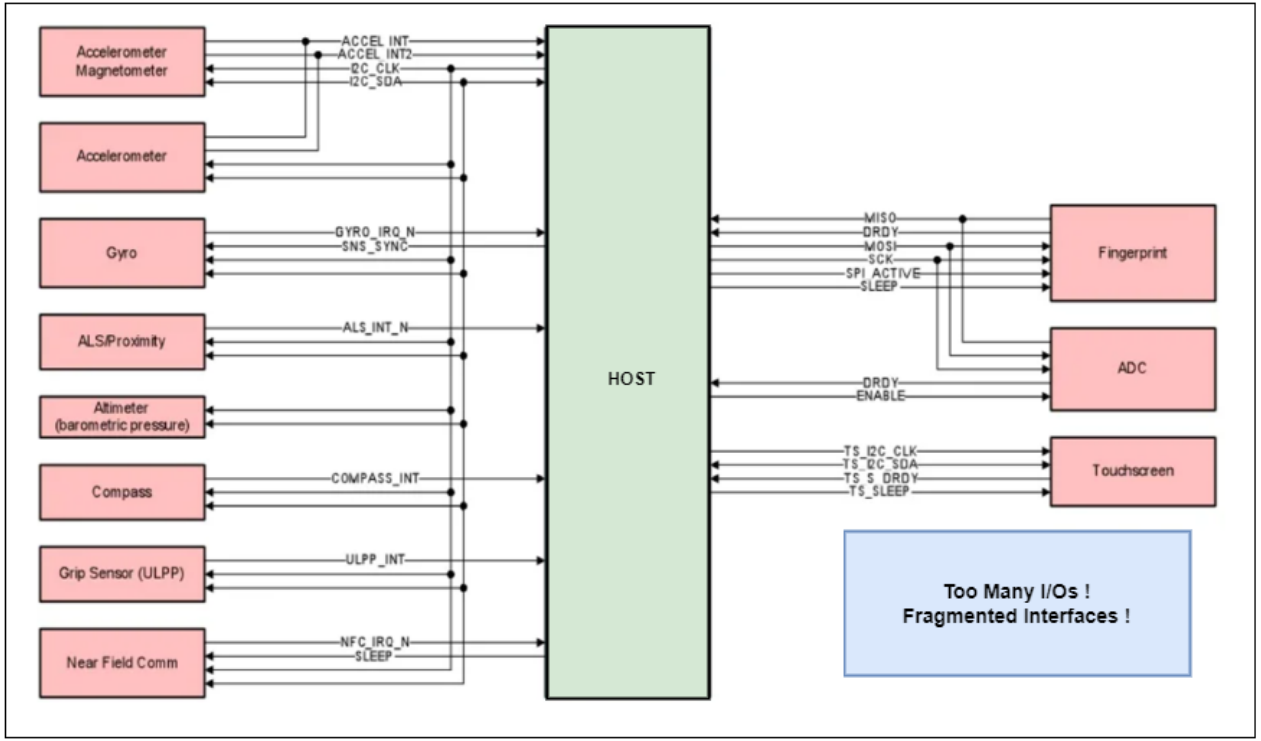




[Image source]

Typical I2C-SPI Sensor-host System

- Multiple Side band signals for interrupts, chip selects, power management
- Standard driver for such fragmented interfaces doesn't exist
- Overall increased package size and added complexity



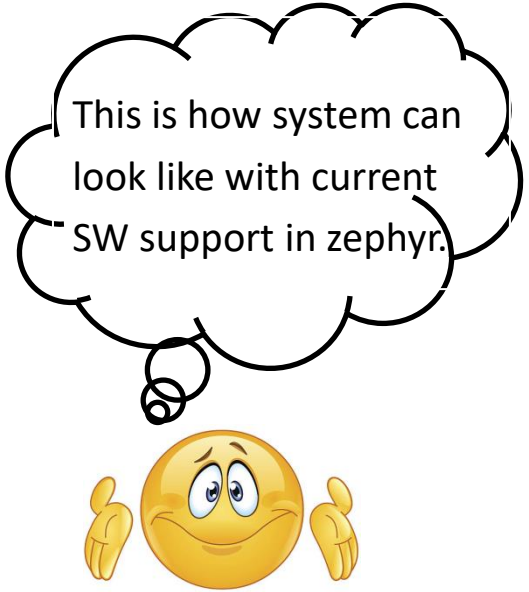
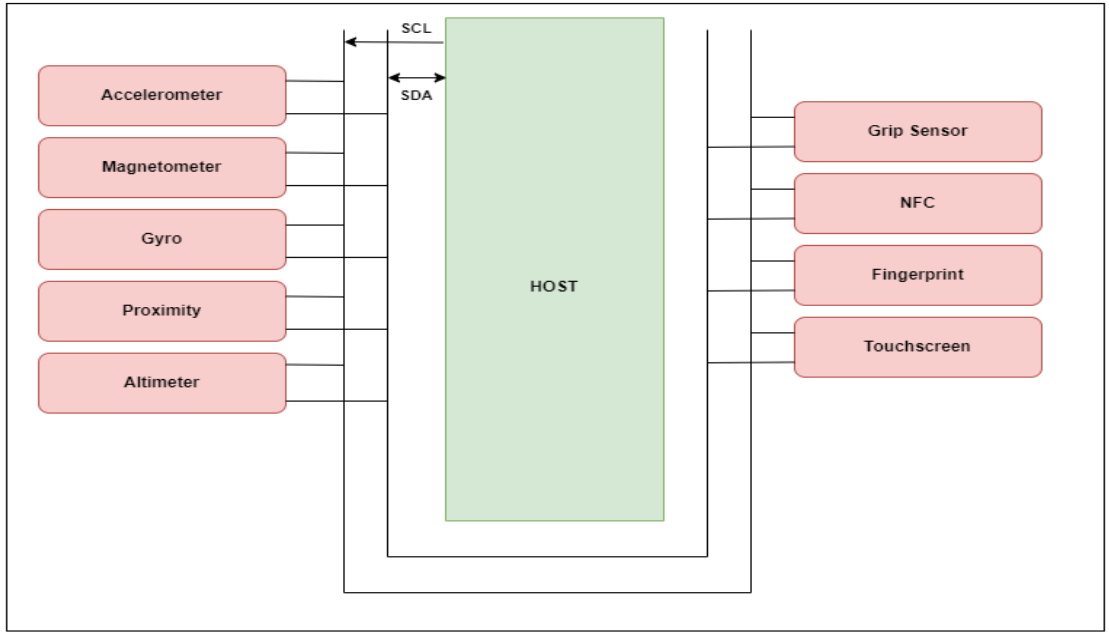


Image source

Multiple Sensor Connectivity Using I3C Interface

- Takes goodness of both I2C (Simplicity & just 2 wires) and SPI (Speed & low power)
- Now have In-Bound Interrupts, Dynamic Address Assignment and High Data Rates
- Old I2C Device support maintained
- Overall less complex and Better power efficiency



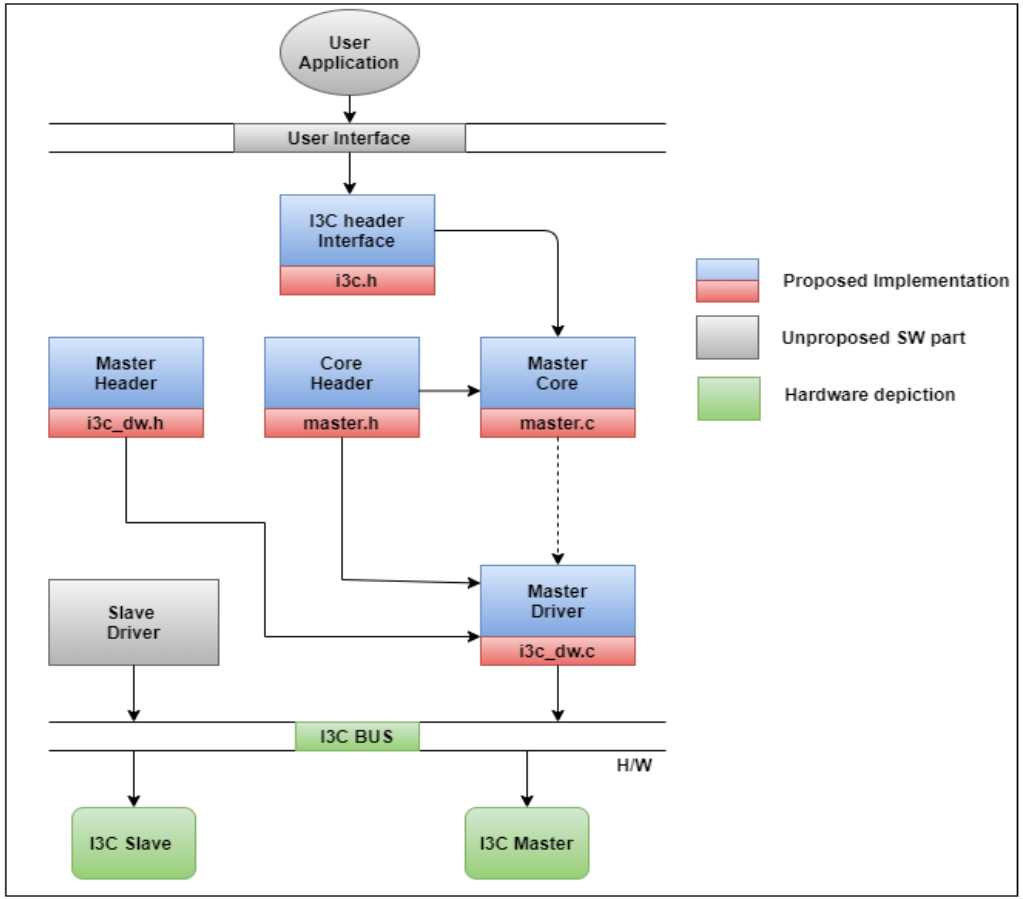
This is how simpler the system can look like with I3C SW support.

Image source

I3C – Proposal for Zephyr RTOS

- Thus we came up with I3C driver support for Zephyr RTOS
- We will be proposing the SW support for I3C Master driver
- The code will be inspired from Linux Kernel
- The driver working has been verified with i3c slaves using zebu emulation platform

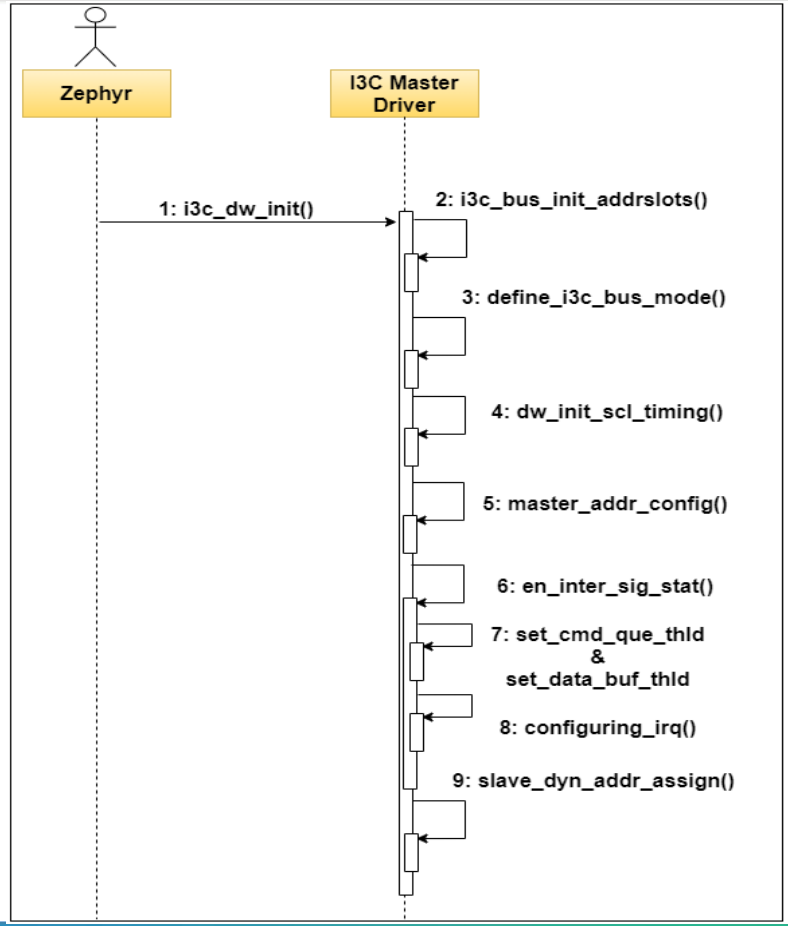
I3C – Proposal for Zephyr RTOS



- I3C SW File Structure Diagram

Driver Software Flow Design

❑ I3C Probe sequence



I3C – Master Driver

Zephyr Project

- When zephyr probes the i3c master driver:

- Then d*w_i3c_config* structure is filled with data from device tree for each master instance.

```
struct dw_i3c_config {
        uint32_t core_clk;
        uint32_t regs;
        uint16_t max_slv_devs;
        uint8_t cmdfifodepth;
        uint8_t datafifodepth;
        void (*configuring_irq)();
 };
```

This struct holds i3c master basic configuration data, here the structure members are :

- core_clk: Device core clock value
- regs: register base address
- max_slv_devs: Max possible slave devices connection
- cmdfifodepth: Depth of the command FIFO
- datafifodepth: Depth of the data FIFO
- configuring_irq: IRQ config function

I3C – Master Driver

- Then d*w_i3c_data* structure is filled with data from device tree for each master instance

```
struct dw_i3c_data {
        struct i3c_master_controller base;
        uint32_t free_pos;
        uint16_t datstartaddr;
        uint16_t dctstartaddr;
        struct dw_slave_info *slave;
        struct k_sem sem_xfer;
        struct k_mutex mt;
        struct dw_i3c_xfer xfer;
};
```

This struct holds master's slave dev related info, here the structure members are :

- base: struct variable holds data for I3C master device
- free_pos: Keeps index of free addr slot
- datstartaddr: DAT start address
- dctstartaddr: DCT start address
- slave: struct ptr that holds Slave device information
- sem_xfer: Semaphore for data manipulation limitations
- mt: mutex for lock on data
- xfer: struct ptr that holds data & related info for data transfer

I3C – Master Driver Probe flow

- Address slots for Dynamic Addressing of slaves are initialized, marking them as free addresses on i3c bus.

```
/* Initializing feasible address slots for DAA */
   i3c_bus_init_addrslots(i3c->base.addrslots);

/* I3C pure, I2C pure or Mix */
    define_i3c_bus_mode(dev);

/* Config timing registers */
   dw_init_scl_timing(dev);
```

- Parameters for SCL timings are initialized on basis of device types connected on i3c bus – i3c only, i2c only or mix.

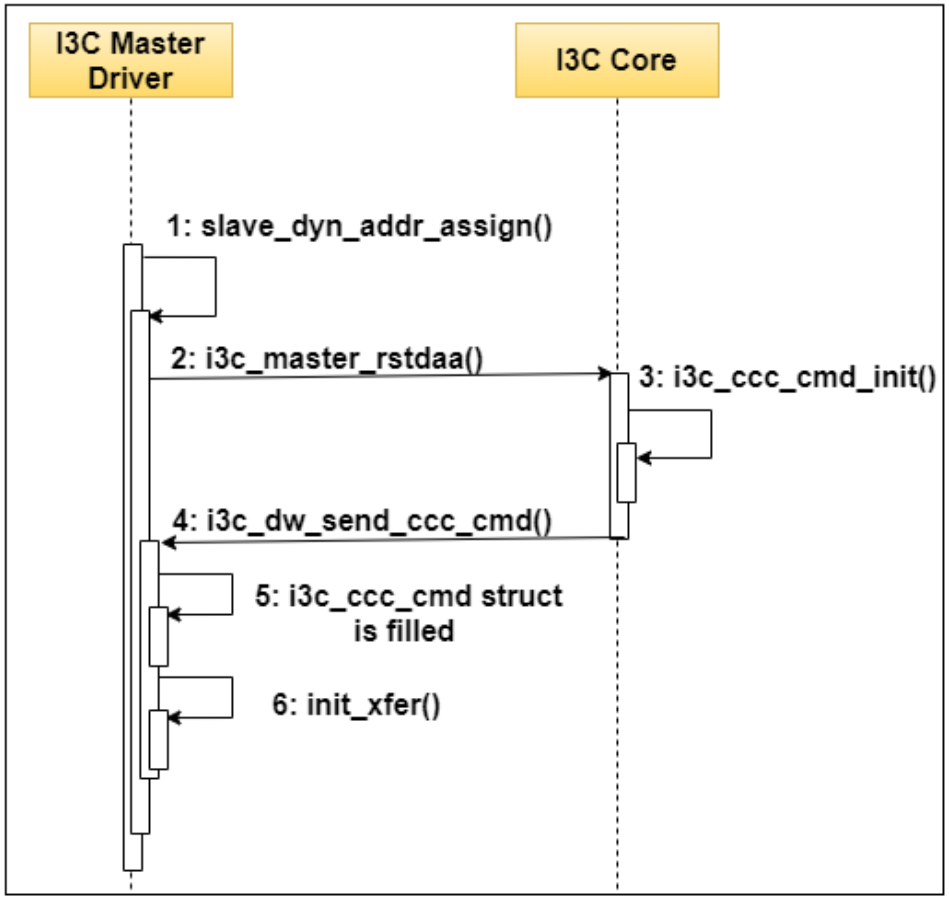
I3C – Master Driver Probe flow

- Timing register settings for SCL are done
- Threshold values of Command Queue, Data Buffer Thresholds are done
- Interrupts are enabled

```
/* enable i3c master */
      sys_write32(sys_read32(DEV_CTRL(config)) | DEV_CTRL_ENABLE, DEV_CTRL(config));
/* Assign dyn addr to slave */
      slave_dyn_addr_assign(dev);
```

- Current driver doesn't support hot join and IBI with payload, features yet
- I3C master is enabled via *ctrl* register & do_daa() is called

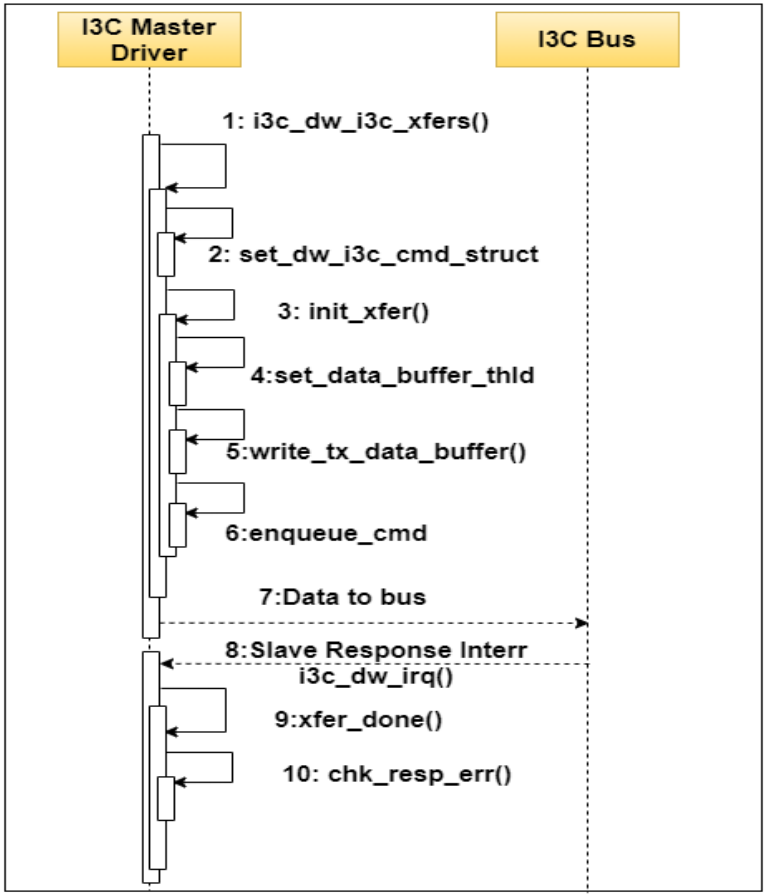
Dynamic Addressing



- The flow diagram here explains the sequence of dynamic addressing during probe.
- *I3c_ccc_cmd* structure takes care of all required command attributes

```
struct i3c_ccc_cmd {
        uint8_t rnw;
        uint8_t id;
        uint8_t addr;
        uint16_t len;
        void* data;
        enum i3c_error_code err;
};
```

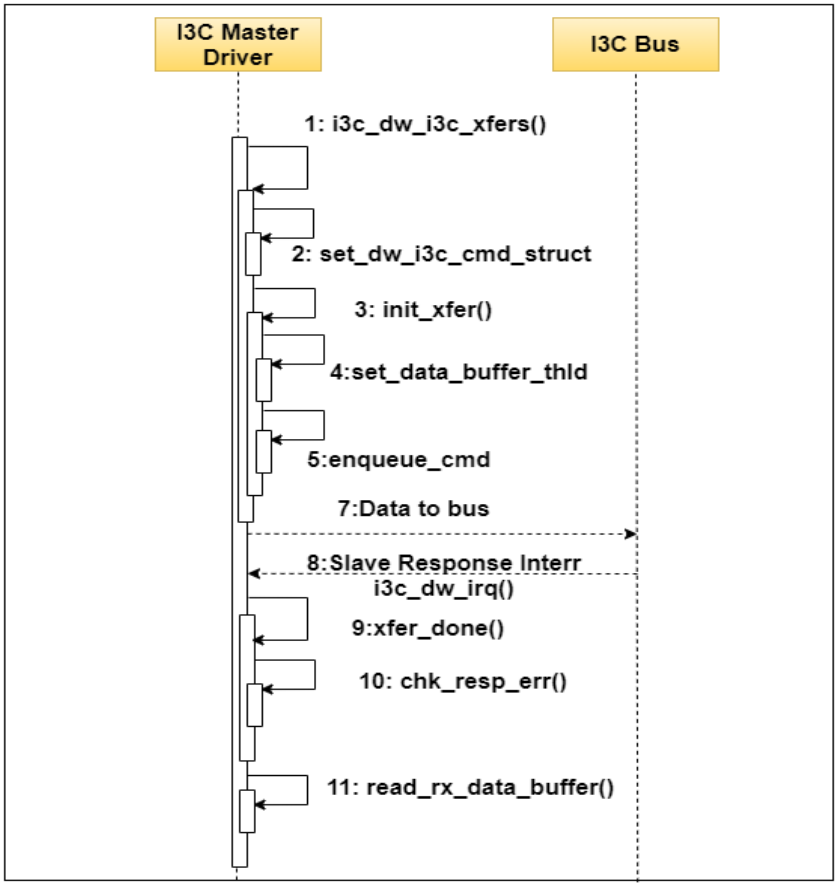
I3C Private Write Sequence



- The flow diagram here explains the sequence of I3C Private write
- *dw_i3c_cmd* structure takes care of attributes required for data transfer

```
struct dw_i3c_cmd {
        uint32_t cmd_lo;
        uint32_t cmd_hi;
        uint16_t tx_len;
        uint16_t rx_len;
        uint8_t error;
        void *buf;
};
```

I3C Private Read Sequence



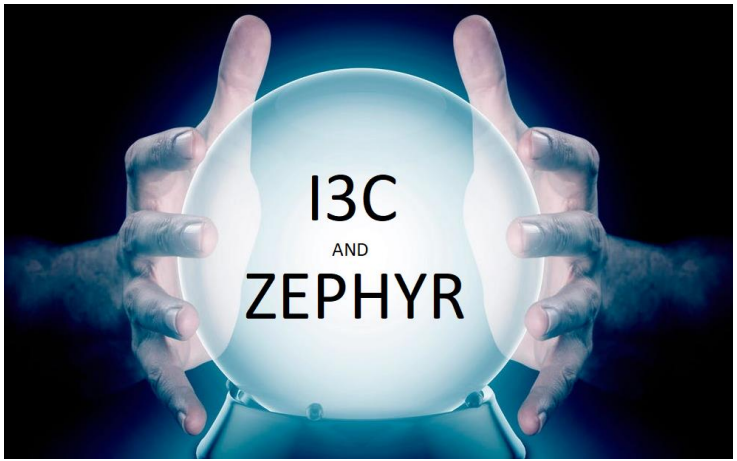
- The flow diagram here explains the sequence of I3C Private read

Conclusion & Future Scope

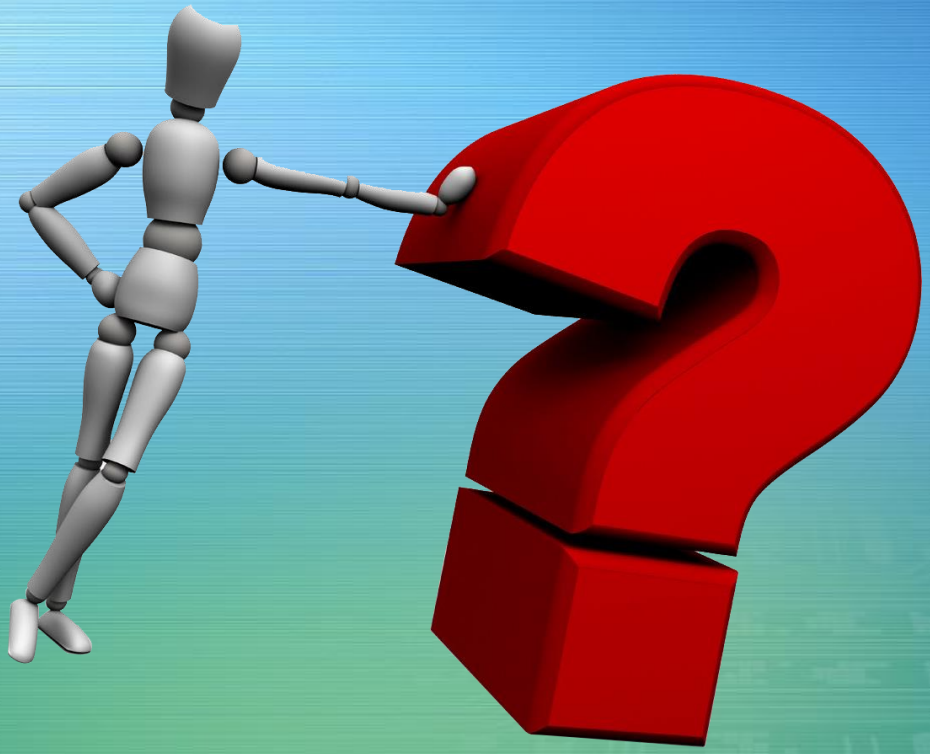
Zephyr® Project

Future scope for i3c software

- ❑ Code corresponding to shared software design in process for mainlining
- ❑ Other I3C Master features already planned for porting to Zephyr
- ❑ Future plans for porting slave driver to Zephyr exists



Any Questions ?



THANK YOU