I3C protocol

Disclaimer

This chapter will focus on aspects that matter to software developers. For everything hardware related (like how things are transmitted on the bus, how collisions are prevented, ...) please have a look at the I3C specification.

This document is just a brief introduction to the I3C protocol and the concepts it brings to the table. If you need more information, please refer to the MIPI I3C specification (can be downloaded here https://resources.mipi.org/mipi-i3c-v1-download).

Introduction

The I3C (pronounced 'eye-three-see') is a MIPI standardized protocol designed to overcome I2C limitations (limited speed, external signals needed for interrupts, no automatic detection of the devices connected to the bus, ...) while remaining power-efficient.

I3C Bus

An I3C bus is made of several I3C devices and possibly some I2C devices as well, but let's focus on I3C devices for now.

An I3C device on the I3C bus can have one of the following roles:

- Master: the device is driving the bus. It's the one in charge of initiating transactions or deciding who is allowed to talk on the bus (slave generated events are possible in I3C, see below).
- Slave: the device acts as a slave, and is not able to send frames to another slave on the bus. The device can still send events to the master on its own initiative if the master allowed it.

I3C is a multi-master protocol, so there might be several masters on a bus, though only one device can act as a master at a given time. In order to gain bus ownership, a master has to follow a specific procedure.

Each device on the I3C bus has to be assigned a dynamic address to be able to communicate. Until this is done, the device should only respond to a limited set of commands. If it has a static address (also called legacy I2C address), the device can reply to I2C transfers.

In addition to these per-device addresses, the protocol defines a broadcast address in order to address all devices on the bus.

Once a dynamic address has been assigned to a device, this address will be used for any direct communication with the device. Note that even after being assigned a dynamic address, the device should still process broadcast messages.

I3C Device discovery

The I3C protocol defines a mechanism to automatically discover devices present on the bus, their capabilities and the functionalities they provide. In this regard I3C is closer to a discoverable bus like USB than it is to I2C or SPI.

The discovery mechanism is called DAA (Dynamic Address Assignment), because it not only discovers devices but also assigns them a dynamic address.

During DAA, each I3C device reports 3 important things:

- BCR: Bus Characteristic Register. This 8-bit register describes the device bus related capabilities
- DCR: Device Characteristic Register. This 8-bit register describes the functionalities provided by the device
- Provisional ID: A 48-bit unique identifier. On a given bus there should be no Provisional ID collision, otherwise the discovery mechanism may fail.

I3C slave events

The I3C protocol allows slaves to generate events on their own, and thus allows them to take temporary control of the bus.

This mechanism is called IBI for In Band Interrupts, and as stated in the name, it allows devices to generate interrupts without requiring an external signal.

During DAA, each device on the bus has been assigned an address, and this address will serve as a priority identifier to determine who wins if 2 different devices are generating an interrupt at the same moment on the bus (the lower the dynamic address the higher the priority).

Masters are allowed to inhibit interrupts if they want to. This inhibition request can be broadcast (applies to all devices) or sent to a specific device.

I3C Hot-Join

The Hot-Join mechanism is similar to USB hotplug. This mechanism allows slaves to join the bus after it has been initialized by the

master.

This covers the following use cases:

- the device is not powered when the bus is probed
- the device is hotplugged on the bus through an extension board

This mechanism is relying on slave events to inform the master that a new device joined the bus and is waiting for a dynamic address.

The master is then free to address the request as it wishes: ignore it or assign a dynamic address to the slave.

I3C transfer types

If you omit SMBus (which is just a standardization on how to access registers exposed by I2C devices), I2C has only one transfer type.

I3C defines 3 different classes of transfer in addition to I2C transfers which are here for backward compatibility with I2C devices.

I3C CCC commands

CCC (Common Command Code) commands are meant to be used for anything that is related to bus management and all features that are common to a set of devices.

CCC commands contain an 8-bit CCC ID describing the command that is executed. The MSB of this ID specifies whether this is a broadcast command (bit7 = 0) or a unicast one (bit7 = 1).

The command ID can be followed by a payload. Depending on the command, this payload is either sent by the master sending the command (write CCC command), or sent by the slave receiving the command (read CCC command). Of course, read accesses only apply to unicast commands. Note that, when sending a CCC command to a specific device, the device address is passed in the first byte of the payload.

The payload length is not explicitly passed on the bus, and should be extracted from the CCC ID.

Note that vendors can use a dedicated range of CCC IDs for their own commands (0x61-0x7f and 0xe0-0xef).

I3C Private SDR transfers

Private SDR (Single Data Rate) transfers should be used for anything that is device specific and does not require high transfer speed.

It is the equivalent of I2C transfers but in the I3C world. Each transfer is passed the device address (dynamic address assigned during DAA), a payload and a direction.

The only difference with I2C is that the transfer is much faster (typical clock frequency is 12.5MHz).

I3C HDR commands

HDR commands should be used for anything that is device specific and requires high transfer speed.

The first thing attached to an HDR command is the HDR mode. There are currently 3 different modes defined by the I3C specification (refer to the specification for more details):

- HDR-DDR: Double Data Rate mode
- HDR-TSP: Ternary Symbol Pure. Only usable on busses with no I2C devices
- HDR-TSL: Ternary Symbol Legacy. Usable on busses with I2C devices

When sending an HDR command, the whole bus has to enter HDR mode, which is done using a broadcast CCC command. Once the bus has entered a specific HDR mode, the master sends the HDR command. An HDR command is made of:

- one 16-bits command word in big endian
- N 16-bits data words in big endian

Those words may be wrapped with specific preambles/post-ambles which depend on the chosen HDR mode and are detailed here (see the specification for more details).

The 16-bits command word is made of:

- bit[15]: direction bit, read is 1, write is 0
- bit[14:8]: command code. Identifies the command being executed, the amount of data words and their meaning
- bit[7:1]: I3C address of the device this command is addressed to
- bit[0]: reserved/parity-bit

Backward compatibility with I2C devices

The I3C protocol has been designed to be backward compatible with I2C devices. This backward compatibility allows one to connect a mix of I2C and I3C devices on the same bus, though, in order to be really efficient, I2C devices should be equipped with 50 ns spike filters.

I2C devices can't be discovered like I3C ones and have to be statically declared. In order to let the master know what these devices

are capable of (both in terms of bus related limitations and functionalities), the software has to provide some information, which is done through the LVR (Legacy I2C Virtual Register).