The BNF grammar and Coding for UCDL

1 The BNF grammar of UCDL

To better present the design of our Unified Chip Description Language (UCDL), in this appendix section, we introduce its BNF grammar. The basic rule of BNF is "<symbol>::=__expression__", where the "symbol" on the left of "::=" is interpreted by the expression on the right. For a better understanding, we first summarize the symbols used in UCDL as follows:

1)"<>" represents the non-terminal symbol. In this paper, its contents are syntactic variables of UCDL.
2)"|" means or. For example, X|Y|Z means the content is selected from one of X, Y, and Z
3)"[]" means that the content is optional. For example, <A>::=<B>[<C>] means that <B> is necessary for the expression of <A>, while <C> is optional and can be defaulted under certain conditions.

Based on this foundation, we introduce the BNF grammar of UCDL in three parts, *i.e.*, pin configuration, data format, and meta operation for chips, as follows.

1.1 Pin Configuration

The statement of pin configuration is guided by the keyword of Pin, and its function is to set pin connections for the plugged chip on LEGO devices. Generally, the statements contain four parameters *i.e.*, pin number, pin type, pin function, and connection properties. Besides, under special bus requirements, the statements will contain some supplement parameters. Its BNF grammar can be summarized as:

▷**UCDL Grammar:** <pin configuration>::=Pin(<pin number>,<pin type>[1],[<pin function>[2]|<fitting voltage>[3]],[<connection properties>[4] — <I2S parameters>[5]],[<UART parameters>[6]])

1) Pin selection: <pin number>::= 1 | 2 | 3 | 4 . . . . . 15 | 16
2) Parameter for pin types: <pin type>[1]::= (SPI | I2C | UART | 1-Wire | RS485 | I2S | PWM | PCM | SMBus) | (VDD | GND | VCC) | (SR | CW | ADC | DAC)
3) Parameter for pin functions: <function>[2]::=DW | DR | CLK | [RST] | [CS] | [Power In] | [Ground]
4) Parameters for voltage sitting: <fitting voltage>[3]::=0.1 | 0.2 | 0.3 $\cdots$ 17.9 | 18.0
5) Parameters for connection properties: <**connection properties**>[4]::= push-pull (in | out) | open-drain (type 1 | type 2) | High-impedance
6) Parameters for I2S Bus: <I2S parameters>[5]::=<operating mode>, <payload size>,<frame size>,<sampling rate>
6.1) <operating mode>::=Standard | Left Justified | Right Justified
6.2) <payload size(frames)>::=1 | 2 | 3 | 4. . . . . 254 | 255
6.3) <frame-size(bit)>::= 16 | 32
6.4) <sampling rate>::=<coefficient><frequency unit>
6.5) <coefficient>::= 1 | 2 | 3 | 4 | 5 | $\cdots$ 998 | 999
6.6) <frequency unit>::= Hz— KHz
7) Parameters for UART Bus: <UART parameters>[6]::=<baud rate>, <payload size>,<parity bit>,<stop bit>,<hardware flow control bit>
7.1)<baud rate(bps)>::= 600 | 1200 | 2400 | 4800 | 9600 | 19200 | 38400 | 76800 | 115200
7.2) <payload size(byte)>::=1 | 2 | 3 | 4. . . . . 254 | 255
7.3) <parity bit>::=0 | 1
7.4) <stop bit>::=1 | 1.5 | 2
7.5) <hardware flow control bit>::= 0 | 1

The annotation of above signs [1]~[6] are as follows.

[1]<pin type> defines what kind of signal is transmitted on the target Pin. Its option can be divided into three categories:

- 1) computer buses like SPI, I2C, USART, 1-Wire, RS485, I2S, PWM, PCM, and SMBus;
- 2) direct voltage interaction. For digital signals (*e.g.*, chip select), the user can choose CW for control signal input for a chip, or select SR to read out the control signal for feedback. For analog signals, the user can select ADC for signal input, or select DAC for signal read out;

- 3) constant output, the user can choose VCC, or VDD for analog or digital power supply for a chip, respectively. The user can also select GND to connect the target pin to the ground.

[2]<pin function> defines the detailed functions of a pin if its type is a data bus. Specifically, computer buses usually consist of several different signal lines, *e.g.*, I2C contains a data line and a synchronized clock line. Therefore, when the pin type is a certain bus, it is necessary to further define its detailed function so that the LEGO device can select the required signal line in the bus kernel. In other cases (*i.e.*, direct voltage interaction and constant output), the signal on the target pin is directly controlled by the gateway instructions, so the detailed function is optional.

[3]<fitting voltage> The UCDL supports analog signal interaction and supply voltage fitting for chips. When the pin is set for analog signal interaction (ADC/DAC), the selected voltage is for the reference voltage for ADC and DAC. The user may typically select 5V as the LEGO boards gives this reference voltage in initial. In addition, when the pin is set for I/O voltage fitting, its range is 3.0-18.0 V as it is achieved by the voltage sifter (MC14504B) on LEGO board with that range.

[4]<connection properties>. defines the electrical characteristics of the pin connections. The choice can be push-pull, open drain, or high impedance.

[5]<I2S parameters>. The I2S bus is dedicated to audio transmission and has some special parameters compared to other conventional digital buses (*e.g.*, I2C, SPI, RS485, 1-wire, etc.). When the selected bus is I2S, it needs to be supplemented with operating mode parameters, payload size, frame size, and sampling rate.

[6]<UART parameters>. The UART bus is also a dedicated bus that has additional configuration parameters. When a pin is set as a UART bus, it is necessary to configure its baud rate, payload size, parity bit, stop bit, and hardware flow control bit for data transmission

1.2 Meta-operation

The statement of meta-operation is guided by one of four keywords: DW (Data Write), DR (Data Read), CW (Control Write), and SR (State Read), which is designed to carry the meta-operation of chip control. Specifically, meta-operation refers to the minimum atomic operation that makes one step change in the chip's internal logic. Therefore, all chip functions can be realized by orchestrating meta-operations in certain timing sequences. The statement of meta-operation contains five variables and parameters, *i.e.*, control keyword, pin number, operation content, processing delay. Its BNF grammar can be summarized as:

▷**UCDL Grammar:** <meta operation>::=<control keyword>[1](<pin number>,<operation content>[2],+<processing delay>[3])

1) Operation definition: <control keyword>[1]::= DW | DR | CW | SR

2) Pin selection: <pin number>::= 1 | 2 | 3 | 4 . . . . . 15 | 16

3) Content for meta-operations: <operation content>[2]::=<DW operation> | <DR operation> | <CW operation> | <SR operation>

3.1) <DW operation>::=0xYY ("YY" is a hexadecimal data, the length is up to 32 bytes.)

3.2) <DR operation>::=(1, X1) | (2, X1, X2) | (3, X1, X2, X3) | · · · | (n, X1, X2 · · · Xn) | (ADC,X1)

3.3) <CW operation>::= (H | L) | (ADC, xxV (0.0≤ xx ≤5.0V))

3.4) <SR operation>::= (H | L) | (DAC, xxV (0.0≤ xx ≤5.0V))

4) Timing control: <processing delay>[3]::=<delay multiples>× <time units>

4.1) <delay multiples>::=1 | 2 | 3 | 4. . . . . 998 | 999

4.2) <time units>::= nS | uS | mS | S

The annotation of above signs [1]∼[3] are as follows.

[1]<control keyword> defines the type of meta-operation. In this parameter, the user has four choices: 1) select DW to write data to the target pin of a chip; 2) select DR to read data from a chip; 3) select CW to send a control signal to the target pin of a chip; 4) select SR to read out voltage states from a pin of the select chip.

[2]<operation content>. This parameter carries the detailed content for the meta-operation. It has 4 categories:

- 1) for data-write (DW) operation, the content is the write-in hexadecimal data with length up to 32 bytes;
- 2) for data-read (DR) operation, the content has two subcategories: a), for digital chips, the content is the length definition for data read out, where the unit is bytes, and the read-out data is marked as "Xi" in each byte. By defining the returned data in each bits, it facilitates data converting for the application (the detail is shown in Section 1.3); b), for analog chips, the content is ADC and Xi, which controls the LEGO device to sample the analog voltage on the target pin by the on-board 8-bit ADC, and return the sampled data.

- 3) for control-write (CW) operation, the content has two subcategories: a), for the digital chip, the input is logic level signal to pull the target pin to logic high (H) or logic low (L); b), for the analog chip, the input is an analog voltage, so the content is DAC (Digital to Analog Converter) and the target voltage. It controls LEGO device to schedule the onboard DAC and output target analog voltage to the target pin at a range from 0-5V.
- 4) for State-read (SR) operation, the content is logic high (H) or logic low (L), for example, SR (4, H) means to check the state of pin 4. Once its outputs turn to logic high (H), it will directly drive the LEGO device to upload the states to the gateway, which indicates the gateway to make subsequent operations.

[3]<Timing control>. This parameter controls the operation interval between individual meta-operations, which is designed to give enough time for the chip to process.

1.3 Data Format Definition (data converting)

The statement of meta-operation is guided by the keyword of DF, which is designed to convert the raw data of chip output to the application data for users. For example, convert "11100101" to 35.4°C. The LEGO gateway utilizes the QScriptEngine [1] for data conversion, which supports the computing of strings with standard mathematical formulas. Hence, the user can directly write the conversion formula as a string. By this, we designs the grammar for chip-output data converting, its grammar can be summarized as:

▷**UCDL Grammar:** <data converting>::=DF(<type number>, <data type>, <equation string>, <data unit>)
1)<type number>::= Y1 — Y2 $\cdots$ — Ym
2)<data type>::= temperature — pressure — humidity.....
3) <equation string>::=Yi=f(Xi, Xj, ...)
4) <data unit>::= kg — °C — N — psi.....

The annotation of this statement are as follows.

The function of <type number> is to distinguish the types of final data (not raw data) that can be output by the chip. For example, the BME280 chip can output temperature and humidity. To distinguish them, we can set the final temperature data as Y1 and the final humidity data as Y2. The function of <data type> is to define the type of the data, it could be temperature, acceleration, pressure or other physical quantity as the chip sensing. If the chip is not a sensor (*e.g.*, an EEPROM), the data type is null.

<equation string> is the equation to convert the raw data (Xi) to the final application data (Yj). The gateway uses DR instructions to get the raw data($X1$,$X2$, $\cdots$, Xn) by reading the chip. Then uses the QScriptEngine to convert out the final data for IoT applications ($Y1$,$Y2$, $\cdots$, Ym). It should be noted that the raw and final application data may not necessarily correspond to one-to-one because the final data may be calculated from multiple raw data. For example, in ADXL362, the final acceleration data needs to add temperature for compensation. In the <Chip Case Study.PDF>, we give 20 examples of COTS chips with data conversion for them. It indicates that LEGO can support complex data conversions for chip outputs. Finally, the <data unit> defines the unit of the converted data. It is linked to <data type>. For example, if the data type is temperature, the unit is °C; if the data type is acceleration, the unit is g (9.8m/S^2).

In summary, the UCDL has a simple syntax and is directly oriented to chip operation, so it is easy to learn and operate. For a better presentation, we present the standards of COTS chips, as shown in Table 1 to make it readable for users.

2 UCDL Syntax Coding

In this section, we introduce operation process of UCDL statements,*i.e.*, how UCDL meta-operations are lowered to instructions, and how these instructions are lowered to underlying signals.

2.1 Overview

In UCDL programming, each instruction is directly linked to a meta-operations of a chip. For a UCDL statement, the gateway automatically maps its keywords and variables into a predefined code and connects them in series to form an independent instruction. Further, the instruction directly drives the USC circuit on LEGO devices to generate the required signal for chip control. The coding details are presented in Section.2.2.

[1]The site of QScriptEngine: https://doc.qt.io/qt-5/qscriptengine.html

Table 1: Specifications of COTS chips in UCDL

Specifications for Pins		Specifications for chip data converting		
Pin Type	**Description Name[1]**	**Data Type**	**Description Name**	**Data Unit**
Pin for analog power supply	VCC	Temperature	Temp, temperature	°C
Pin for digital power supply	VDD	Relative humidity	RH	%
Pin for push-pull output	Push-pull(output)	Pressure	P,Pre	Psi
Pin for push-pull input	Push-pull(input)	Blood oxygen concentration	SPO2	%
Pin for open-drain connection (type1)	Open-drain (type1)	Local temperature	LT, local temperature	°C
Pin for open-drain connection (type2)	Open-drain (type2)	Remote temperature	RT, remote temperature	°C
Pin for chip select functions	CS	Concentration of PM2.5	PM2.5 level	ppm
Pin for interrupt interactions	Int	Light intensity	light level, light intensity	Lux
Pin for clock signal transmission	Clock	Manufacturer ID	Manufacturer ID, MID	null
Pin for data signal transmission	Data	Concentration of CO2	eCO2	ppm
Pin for chip reset functions	RST	Concentration of TVOC[2]	eTVOC	ppm
Pin for I2C connection	I2C	Intensity of ultraviolet ray	UV Intensity	mW/cm^2
Pin for SPI connection	SPI	Rotation angle	Angle	°
Pin for I2S connection	I2S	Air alcohol concentration	Alcohol	ppm
Pin for SMBus cnnection	SMBus	Concentration of dust	DustDensity	ppm
Pin for 1-wire cnnection	1-Wire	Acceleration	Acceleration	g, m/s^2
Pin for UART connection	UART	Concentration of combustible gas	Gas	ppm
Pin for analog voltage output	ADC	Fluid flow velocity	Flow velocity	m/S
Pin for analog voltage input	DAC	Magnetic field intensity	Magnetic field	A/m

[1] Description name represents the specific name in UCDL syntax.
[2] TVOC means Total Volatile Organic Compounds

2.2 Coding Details

In a description file, the instructions for pin configurations (keywords: *PIN*) and logic control (keywords: *DW*, *DR*, *CW*, *SR*) are converted into gateway instructions. In contrast, the instructions for data formats decoding (keywords: *DF*) is only running on the gateway, so it does not need a coding map, and the gateway directly extracts the content for data display, we discuses it in the end of this section. The encoding scheme of the 5 gateway instructions is as follows:

PIN instruction(keywords: *PIN*): The coding for Pin instruction has the main four cases:

- 1) Generally, the code format for pin configuration instruction consists of six fields, *i.e.*, 4-bit device code[2], 4-bit keyword code, 4-bit pin number code, 4-bit pin type code, 4-bit pin function code, and 5-bit connection type code. The general format fit for most type of digital buses, *i.e.*, I2C, SPI, 1-Wire, SMBus, etc.

- 2) When the pin is set for voltage fitting or analog signal interaction (with pin types of ADC, DAC and VDD, VCC), in its format, the 4-bits pin function code is changed to 8-bit voltage setting code. Hence, its code format is: 4-bit device code, 4-bit keyword code, 4-bit pin number code, 4-bit pin type code, 8-bit voltage setting code, and 5-bit connection type code.

- 3) When the pin is set for I2S connection, in its format, the 4-bits pin function code is changed to 24-bits I2S parameters. The I2S parameter contains 2-bit operating mode code, 8-bit payload size code, 2-bit frame size code, and 12-bit sample rate code (consists of 10-bit coefficient and 2-bit unit). Hence, its code format is: 4-bit device code, 4-bit keyword code, 4-bit pin number code, 4-bit pin type code, 24-bit I2S parameter code, and 5-bit connection type code.

- 3) When the pin is set for UART connection, in its format, the 4-bits pin function code is changed to 18-bits UART parameters. The UART parameter contains 4-bit baud rate code, 8-bit payload size code, 2-bit parity bit code, 2-bit stop bit code, and 2-bit hardware flow control bit code. Hence, its code format is: 4-bit device code, 4-bit keyword code, 4-bit pin number code, 4-bit pin type code, 18-bit UART parameter code, and 5-bit connection type code.

DW instruction(keywords: *DW*): The code format for data write instruction consists of four fields, *i.e.*, 4-bit device code, 4-bit keyword code, 4-bit pin number code, and n-bit write-in data code (the data to write in the target chip).

DR instruction(keywords: *DR*) The code format for data read instruction consists of four fields, *i.e.*, 4-bit device code, 4-bit keyword code, 4-bit pin number code, and 4-bit read length definition code (number of Bytes read out from the target chip).

[2]In the initial stage, the system uses a 4-bit device code and a 4-bit pin address for the LEGO device. Hence a gateway can support up to 16 LEGO devices in its network.

Table 2: UCDL Syntax Coding

	Keywords	Code	Description
Coding For Keywords	PIN	1000	Set pin functions for a chip
	DF	1001	Define data format for a chip
	DW	1010	Write data to a chip
	DR	1011	Read data from a chip
	CW	1100	Send a control signal to a chip
	SR	1101	Read a status signal from a chip
	Pin Types	Code	Description
Coding For Pin Types	I2C	0001	Set Connections for I2C Bus
	SPI	0010	Set Connections for SPI Bus
	USART	0011	Set Connections for USART Bus
	1-Wire	0100	Set Connections for 1-Wire Bus
	RS485	0101	Set Connections for RS485 Bus
	I2S	0110	Set Connections for I2S Bus
	PWM	0111	Set Connections for PWM Bus
	PCM	1000	Set Connections for PCM Bus
	CW	1001	Set Control-Write Pin
	SR	1010	Set State-Read Pin
	Pin Functions	Code	Description
Coding For Pin Functions	CLK	0010	Clock signal transmission
	VCC	0011	Power supply (Analog)
	DW	0100	Data input(write)
	DR	0101	Data output(read)
	DW/DR	0110	Data Write/Read
	VDD	0111	Power supply (Digital)
	GND	1000	Ground
	Connection Types	Code	Description
Coding For Connection Types	Push-Pull (output)	11000	Set connection as input with push-pull
	Push-Pull (input)	00100	Set connection as output with push-pull
	Open-Drain (type 1)	01010	Set connection as type 1 of open-drain
	Open-Drain (type 2)	10001	Set connection as type 2 of open-drain
	High Impedance	00000	Set connection as high impedance

CW instruction(keywords: *CW*): The code format for control write instruction consists of four fields, *i.e.*, 4-bit device code, 4-bit keyword code, 4-bit pin number code, and 1-bit control flag code (to set the target pin of a chip to logic high (flag=1) or logic low (flag=0)).

SR instruction: The code format for state read instruction consists of three fields, *i.e.*, 4-bit device code, 4-bit keyword code and 4-bit pin number code.

The code mapping of the keywords and variables of above five instructions is shown in Table 2. For instance, instruction $'0011-1000-0101-0010-11000'$ means to set the 5th pin of the LEGO device with ID 3 to SPI output for data writing, and the connection type is push-pull. When the device receives this instruction, the Universal Signal Conversion (USC) circuit acts accordingly to build up the required connections.