Лабораторна робота №5-6

Дослідження цифро-аналогового перетворення за допомогою DAC0808 та Atmega128

Тема: Дослідження особливостей цифро-аналогового перетворення в мікрочіпі Atmega128.

Мета: Дослідити методи та засоби цифро-аналогового перетворення.

Цифро-аналоговий перетворювач (ЦАП) - пристрій для перетворення цифрового (зазвичай двійкового) коду на аналоговий сигнал (струм, напруга або заряд). Цифро-аналогові перетворювачі є інтерфейсом між дискретним цифровим світом та аналоговими сигналами. Сучасні ЦАП створюються за напівпровідниковими технологіями у вигляді інтегральної схеми.

ЦАП застосовується завжди у телекомунікаційних системах та системах управління. Наприклад:

- У системах відтворення аудіо;
- У дисплеях;
- Формування інформаційного сигналу для змішувачів та керованих генераторів;
- У системах керування двигуном;
- У системах прямого цифрового синтезу (DDS – Direct Digital Synthesizer);

Аналого-цифровий перетворювач (АЦП) здійснює зворотну операцію.

Загальні технічні характеристики

- Розрядність. Визначає кількість рівнів аналогового сигналу, який може відтворювати ЦАП. Для N-розрядного ЦАП число рівнів аналогового сигналу дорівнює 2N (включаючи значення для нульового коду);
- Напруга живлення;

Статичні характеристики:

- Статична характеристика перетворення – це залежність значення вихідного сигналу ЦАП значення вхідного коду.
- Статична нелінійність. Для опису статичної нелінійності використовують дві величини: диференціальна нелінійність (DNL) та інтегральна нелінійність (INL).
- Монотонність. Одна з найважливіших характеристик ЦАП, яка говорить про те, що при збільшенні коду аналогового сигналу також збільшується. Унарна архітектура гарантує монотонність. Для бінарної архітектури монотонність не гарантується.
- Зміщення нуля.
- Помилка посилення.

Динамічні характеристики:

- Швидкодія. Визначається як максимальна частота, з якою можна змінювати код на вході ЦАП, отримуючи коректний результат на його виході. Вимірюється у «вибірках/сек» чи герцах. Може називатися частота дискретизації або максимальна частота зміни вхідного коду.

- SNR (відношення сигнал/шум). Вважається як відношення потужності гармонійного сигналу, що відновлюється, до суми потужностей всіх інших гармонік у спектрі вихідного сигналу, крім кратних, і виражається в децибелах.
- SFDR (динамічний діапазон, вільний від паразитних складових). Вважається як відношення амплітуди гармонійного сигналу, що відновлюється, до амплітуди найбільшої гармоніки в спектрі вихідного сигналу, також виражається в децибелах. Цю характеристику також називають " динамічною лінійністю " .
- Споживана потужність.

У послідовних ЦАП вхідний код перетворюється на аналоговий сигнал порозрядно. При цьому для перетворення всіх розрядів використовується та сама схема, що значно спрощує пристрій, проте швидкість перетворення в таких обернено пропорційна розрядності. Не варто плутати спосіб перетворення і вхідний інтерфейс пристрою: на вхід ЦАП послідовного вхідний код може подаватися як послідовно, так і паралельно. До послідовних ЦАП можна віднести такі види:

- **Широтно-імпульсний модулятор** – найпростіший тип ЦАП. Стабільний джерело струму або напруги періодично включається на час, пропорційне цифровому коду, що перетворюється, далі отримана імпульсна послідовність фільтрується аналоговим фільтром нижніх частот. Такий спосіб часто використовується для управління швидкістю електромоторів, а також стає популярним у Hi-Fi аудіотехніці;
- **Циклічний ЦАП** (cyclic DAC);
- **Конвеєрний ЦАП** (pipeline DAC);

Звуковий ЦАП зазвичай отримує на вхід цифровий сигнал імпульсно-кодової модуляції (PCM, pulse-code modulation). Завдання перетворення різних стислих форматів PCM виконується відповідними кодеками.

Архітектури

Архітектура ЦАП це спосіб формування вихідного сигналу на функціональному рівні. Інакше кажучи, це опис того, на суму яких чисел буде розкладатися значення вихідного сигналу. Вихідний сигнал формується за допомогою елементів, що зважують, кожен з яких відповідає за свою "порцію" вихідного аналогового сигналу. Розрізняють такі архітектури за набором значень елементів, що зважують:

Бінарна архітектура. Співвідношення двох сусідніх елементів, що зважують, дорівнює 2. Тобто вихідний сигнал формується так само, як це відбувається в двійковій системі числення. Відповідно, ваги елементів, що формують вихідний сигнал, в нормованому вигляді, дорівнюватимуть 1, 2, 4, 8, 16 і т. д. Управління зважуючими елементами здійснюється бінарним кодом.

Унарна архітектура. Співвідношення двох сусідніх елементів, що зважують, дорівнює 1. Тобто вихідний сигнал формується так само, як це відбувається в унарній системі числення. Відповідно, ваги всіх елементів, у нормованому вигляді, дорівнюють 1. Управління здійснюється унарним або унітарним кодом.

Архітектура Фібоначчі. Ваги елементів є послідовністю чисел Фібоначчі. Вихідний сигнал формується так само, як це відбувається в Фібоначчієвій системі числення.

Крім того, існує поняття сегментної архітектури, яка передбачає поділ вхідного коду на кілька груп. Як правило, дві. Кожна група опрацьовується незалежно своїм сегментом. Вихідні сигнали всіх сегментів комбінуються утворюючи вихідний сигнал ЦАП. Найчастіше зустрічається наступна конфігурація сегментної архітектури: молодші розряди обробляються сегментом, побудованим по бінарної архітектури, старші розряди - сегментом, побудованим по унарної архітектурі.

Типи зважувальних елементів та способи формування ваги

Цифро-аналогові перетворювачі незалежно від архітектури можуть використовувати як елемент, що зважує аналоговий сигнал, такі типи компонентів: конденсатори, резистори та джерела струму.

Конденсатори. Даний тип зважувальних елементів у разі застосування в бінарній архітектурі може мати номінали, що відрізняються у сусідніх елементів в 2 рази, або мати номінали 1 і 2 і формувати сходовий ланцюг C-2C.

Резистори. Цей тип зважувальних елементів має самі принципи побудови, що конденсатори. З іншого боку, існують реалізації подібних структур з урахуванням не резисторів, а транзисторів, які у ролі резисторів. Такі ланцюги називаються M-2M.

Джерела струму. Це, зазвичай, транзистор як насичення. Використання даних типів елементів, що зважують, дозволяє обійтися без буферів, які необхідні для інших типів елементів, що зважують.

Для формування ваги елемента, що зважує, є наступні способи:

Масштабування номіналів. Застосовується до будь-якого типу елементів, що зважують. З погляду напівпровідникової технології це завжди еквівалентно масштабування розмірів елементів.

Використання сходової структури. Застосовується тільки до ємнісних і резистивних елементів, що зважують. Залежно від типу елемента, що зважує, такі структури отримують назви R-2R, C-2C або M-2M (замість резисторів використовуються транзистори).

Зміна напруги усунення. Застосовується лише до джерел струму. Зміна напруги зміщення може відбуватися як за допомогою ланцюга, що перебудовується формування напруги зміщення, так і за допомогою інжекції заряду на плаваючий затвор. Останнє застосовується лише для спеціальних технологій, що передбачають формування плаваючого затвора у транзистора. Як правило, це технології, призначені для виготовлення енергонезалежної пам'яті.

Структури резистивних та ємнісних паралельних ЦАП

ЦАП зважувального типу, в якому кожному біту двійкового коду, що перетворюється, відповідає резистор або джерело струму, підключений на загальну точку підсумовування. Сила струму джерела (провідність резистора) пропорційна ваги біта, якому він відповідає. Таким чином, усі ненульові біти коду підсумовуються з вагою. Зважуючий метод один із найшвидших, але йому властива низька точність через необхідність наявності набору безлічі різних прецизійних джерел або резисторів та непостійного імпедансу. Тому зважуючі ЦАП мають розрядність трохи більше восьми біт.

ЦАП сходового типу (ланцюгова R-2R-схема). У R-2R-ЦАП значення створюються у спеціальній схемі, що складається з резисторів з опорами R і 2R, що називається матрицею постійного імпедансу, яка має два види включення: пряме - матриця струмів та інверсне - матриця напруг. Застосування однакових резисторів дозволяє суттєво поліпшити точність порівняно зі звичайним ЦАП, що зважує, так як порівняно просто виготовити набір прецизійних елементів з однаковими параметрами. ЦАП типу R-2R дозволяють відсунути обмеження розрядності. З лазерним припасуванням плівкових резисторів, розташованих на одній підкладці гібридної мікросхеми, досягається точність 20-22 біта. Основна затримка перетворення визначається затримкою в операційному підсилювачі, тому він повинен мати максимальну швидкодію.

Типова швидкодія ЦАП - одиниці мікросекунд і нижче до одиниць наносекунд.

ASCII printable characters (character code 32-127)

DEC	OCT	HEX	BIN	Symbol
32	040	20	00100000	SP
33	041	21	00100001	!
34	042	22	00100010	"
35	043	23	00100011	#
36	044	24	00100100	$
37	045	25	00100101	%
38	046	26	00100110	&
39	047	27	00100111	'
40	050	28	00101000	(
41	051	29	00101001	)
42	052	2A	00101010	*
43	053	2B	00101011	+
44	054	2C	00101100	,
45	055	2D	00101101	-
46	056	2E	00101110	.
47	057	2F	00101111	/
48	060	30	00110000	0

DEC	OCT	HEX	BIN	Symbol
49	061	31	00110001	1
50	062	32	00110010	2
51	063	33	00110011	3
52	064	34	00110100	4
53	065	35	00110101	5
54	066	36	00110110	6
55	067	37	00110111	7
56	070	38	00111000	8
57	071	39	00111001	9
58	072	3A	00111010	:
59	073	3B	00111011	;
60	074	3C	00111100	<
61	075	3D	00111101	=
62	076	3E	00111110	>
63	077	3F	00111111	?
64	100	40	01000000	@
65	101	41	01000001	A
66	102	42	01000010	B
67	103	43	01000011	C
68	104	44	01000100	D
69	105	45	01000101	E
70	106	46	01000110	F
71	107	47	01000111	G
72	110	48	01001000	H

DEC	OCT	HEX	BIN	Symbol
73	111	49	01001001	I
74	112	4A	01001010	J
75	113	4B	01001011	K
76	114	4C	01001100	L
77	115	4D	01001101	M
78	116	4E	01001110	N
79	117	4F	01001111	O
80	120	50	01010000	P
81	121	51	01010001	Q
82	122	52	01010010	R
83	123	53	01010011	S
84	124	54	01010100	T
85	125	55	01010101	U
86	126	56	01010110	V
87	127	57	01010111	W
88	130	58	01011000	X
89	131	59	01011001	Y
90	132	5A	01011010	Z
91	133	5B	01011011	[
92	134	5C	01011100	\
93	135	5D	01011101	]
94	136	5E	01011110	^
95	137	5F	01011111	_
96	140	60	01100000	`

DEC	OCT	HEX	BIN	Symbol
97	141	61	01100001	a
98	142	62	01100010	b
99	143	63	01100011	c
100	144	64	01100100	d
101	145	65	01100101	e
102	146	66	01100110	f
103	147	67	01100111	g
104	150	68	01101000	h
105	151	69	01101001	i
106	152	6A	01101010	j
107	153	6B	01101011	k
108	154	6C	01101100	l
109	155	6D	01101101	m
110	156	6E	01101110	n
111	157	6F	01101111	o
112	160	70	01110000	p
113	161	71	01110001	q
114	162	72	01110010	r
115	163	73	01110011	s
116	164	74	01110100	t
117	165	75	01110101	u
118	166	76	01110110	v
119	167	77	01110111	w
120	170	78	01111000	x

DEC	OCT	HEX	BIN	Symbol
121	171	79	01111001	y
122	172	7A	01111010	z
123	173	7B	01111011	{
124	174	7C	01111100	|
125	175	7D	01111101	}
126	176	7E	01111110	~
127	177	7F	01111111	DEL

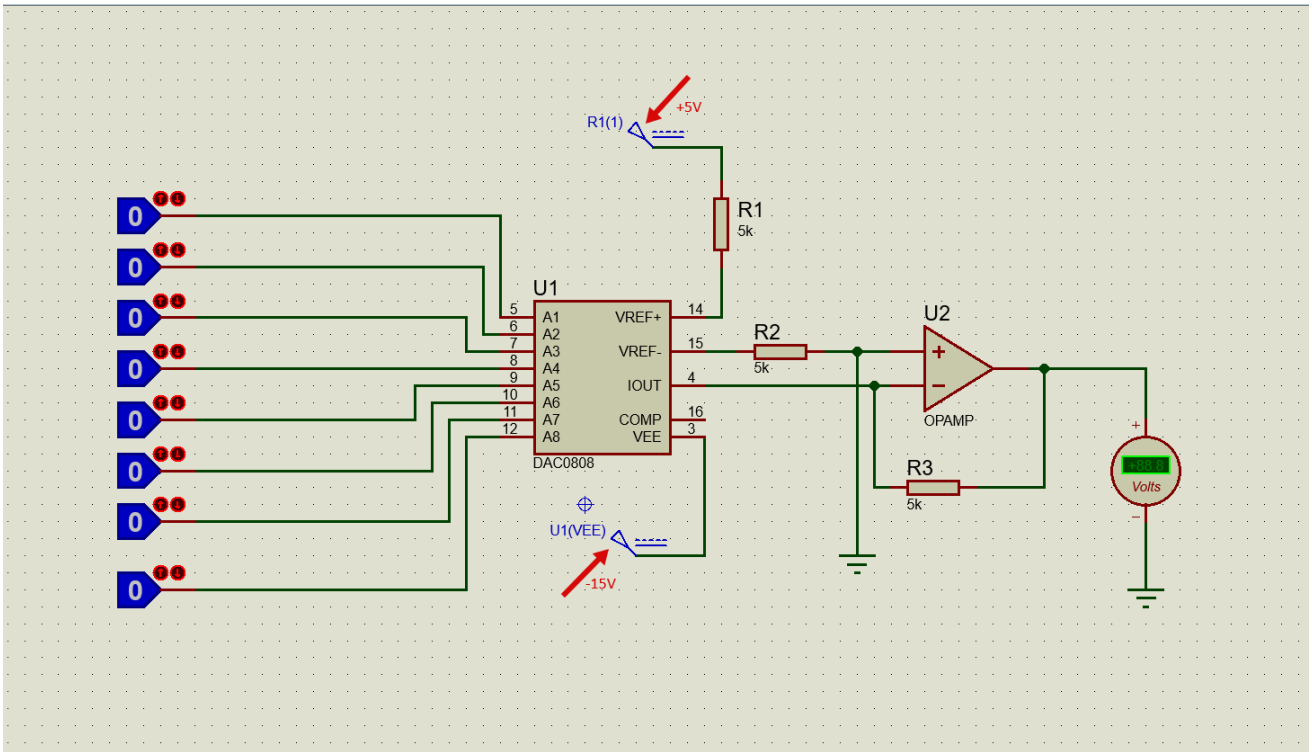

Рис.5.1 – Схема підключення ЦАП.

5.2 Порядок виконання лабораторної роботи

1. Ознайомитись з теоретичними відомостями;
2. Побудувати схему зображену на рисунку 5.1;
3. За допомогою активації відповідних логічних станів (бітів) отримати значення вихідної напруги відповідно до варіанту;
4. Підключити ЦАП до контролера Atmega128;
5. За допомогою кодування ASCII закодувати своє прізвище, ініціали та групу та перетворити в аналоговий сигнал;
6. Підключити на вихід ЦАПу осцилограф та перевірити роботу;

7. Додати до схеми ще один контролер та налаштувати роботу АЦП на прийом інформації з ЦАП;
8. Розкодувати отриманий сигнал за допомогою АЦП (п.5) та вивести на дисплеї;
9. Зробити висновки .

5.3 Зміст звіту

1. Найменування і мета роботи.
2. Отриманий результат роботи схеми відповідно рисунку 5.1.
3. Схема підключення ЦАП до Atmega128.
4. Данні з осцилографу(п.5).
5. Результат розшифрування розкодованого повідомлення на дисплеї.
6. Висновки по роботі.

5.4 Контрольні запитання

1. Що таке ЦАП (цифровий аналоговий перетворювач) і як він працює?
2. Які типи ЦАП існують і в чому їхні основні відмінності?
3. Як налаштувати осцилограф для перевірки сигналу на виході ЦАП?
4. Яким чином вивести розкодовану інформацію на дисплей (наприклад, на дисплеї LM016L)?
5. Які можливі джерела помилок в роботі ЦАП та АЦП?
6. Як впливає точність ЦАП і АЦП на результат виконання завдання?

Варіант	1,11,21	2,12,22	3,13,23	4,14,24	5,15,25	6,16,26	7,17,27	8,18,28	9,19,29	10,20,30
Значення вихідної наруги	0.5V	1V	1.5V	2V	2.5V	3V	3.5V	4V	4.5V	5V

```
#include <avr/io.h>
#include <util/delay.h>
#include <stdio.h>

// Ініціалізація порту для ЦАП
void DAC_init(void) {
	DDRD = 0xFF;  // Установлення всіх ліній порту D як вихідних
}

// Вивід 8-бітного значення на ЦАП
void DAC_write(uint8_t value) {
	PORTD = value;  // Вивід значення на порт D
}

// Функція для дзеркального відображення 8-бітового значення
uint8_t mirror_bits(uint8_t value) {
	uint8_t mirrored = 0;
	for (int i = 0; i < 8; i++) {
		if (value & (1 << i)) {
			mirrored |= (1 << (7 - i));
		}
	}
	return mirrored;
}

// Функція для кодування слова у 8-бітові значення і передачі через ЦАП
void transmit_word_as_signal(const char* word) {
	while (*word) {
		uint8_t encoded_char = (uint8_t)(*word);  // Перетворення символу у 8-бітове
значення

		// Дзеркальне відображення 8-бітового значення
		encoded_char = mirror_bits(encoded_char);

		// Вивід значення на ЦАП
		DAC_write(encoded_char);

		// Затримка для демонстрації аналогового сигналу
		_delay_ms(100);

		word++;
	}
}

int main(void) {
	DAC_init();  // Ініціалізація ЦАП

	const char* message = "Hello, world! <";  // Повідомлення для передачі
// “<” – реалізована логіка очищення дисплею


	while (1) {
		transmit_word_as_signal(message);
	}
}
```

```c
#define F_CPU 1000000UL // Частота мікроконтролера 1 МГц
#include <avr/io.h>
#include <util/delay.h>
// Піни для підключення дисплея
#define RS PB0
#define EN PB1
#define D4 PB2
#define D5 PB3
#define D6 PB4
#define D7 PB5
// Функції для керування дисплеєм
void LCD_Command(unsigned char cmd);
void LCD_Char(unsigned char data);
void LCD_Init(void);
void LCD_String(const char *str);
void LCD_Clear(void);
// Встановлення порту і піна для дисплея
void LCD_Port(unsigned char data) {
	if (data & 1) PORTB |= (1<<D4);
	else PORTB &= ~(1<<D4);
	if (data & 2) PORTB |= (1<<D5);
	else PORTB &= ~(1<<D5);
	if (data & 4) PORTB |= (1<<D6);
	else PORTB &= ~(1<<D6);
	if (data & 8) PORTB |= (1<<D7);
	else PORTB &= ~(1<<D7);
}
void LCD_Command(unsigned char cmd) {
	// Відправляємо команду (RS=0)
	PORTB &= ~(1<<RS);
	// Передаємо старші 4 біти
	LCD_Port(cmd >> 4);
	PORTB |= (1<<EN);
	_delay_ms(1);
	PORTB &= ~(1<<EN);
	// Передаємо молодші 4 біти
	LCD_Port(cmd);
	PORTB |= (1<<EN);
	_delay_ms(1);
	PORTB &= ~(1<<EN);
}
void LCD_Char(unsigned char data) {
	// Відправляємо дані (RS=1)
	PORTB |= (1<<RS);
	// Передаємо старші 4 біти
	LCD_Port(data >> 4);
	PORTB |= (1<<EN);
	_delay_ms(1);
	PORTB &= ~(1<<EN);
	// Передаємо молодші 4 біти
	LCD_Port(data);
	PORTB |= (1<<EN);
	_delay_ms(1);
	PORTB &= ~(1<<EN);
}
void LCD_Init(void) {
	// Налаштовуємо пін як вихід
	DDRB |= (1<<RS) | (1<<EN) | (1<<D4) | (1<<D5) | (1<<D6) | (1<<D7);
	// Початкова ініціалізація
	LCD_Command(0x02); // Перемикання в 4-бітний режим
	LCD_Command(0x28); // 2 рядки, 5x8 точок
	LCD_Command(0x0C); // Увімкнути дисплей, без курсора
	LCD_Command(0x06); // Інкрементувати курсор
```

```
	LCD_Command(0x01); // Очистити дисплей
	_delay_ms(2);
}
void LCD_String(const char *str) {
	while (*str) {
		LCD_Char(*str++);
	}
}
void LCD_Clear(void) {
	LCD_Command(0x01); // Команда очищення дисплея
	_delay_ms(2);
}

// Ініціалізація АЦП
void adc_init() {
	ADMUX = (1<<REFS0);  // Використовуємо внутрішнє опорне напруження AVCC
	ADCSRA = (1<<ADEN) | (1<<ADPS2) | (1<<ADPS1) | (1<<ADPS0);  // Вмикаємо АЦП та
встановлюємо преддільник на 128
}

// Читання значення з АЦП
uint16_t adc_read(uint8_t channel) {
	channel &= 0x07;                // Обмежуємо номер каналу до діапазону [0..7]
	ADMUX = (ADMUX & 0xF8) | channel;  // Вибираємо канал (ADC0 на PF0)
	ADCSRA |= (1<<ADSC);            // Починаємо перетворення
	while (ADCSRA & (1<<ADSC));     // Чекаємо завершення перетворення
	return ADC;                     // Повертаємо результат
}

int decode_signal(void) {
	uint8_t decoded_value;
	while (1) {
		uint16_t adc_result = adc_read(0); // Зчитування значення з каналу 0
		decoded_value = adc_result >> 2; // Перетворення 10-бітного значення у 8-бітне
		// Відображення значення або подальша обробка
		// Наприклад, передача у віртуальний термінал через UART
		return decoded_value;
	}
}

int main(void) {
	// Ініціалізація дисплея
	LCD_Init();
	adc_init();

	while (1) {
		uint16_t adc_value = adc_read(0); // Читаємо значення з каналу 0 (PF0)
		uint8_t decoded_char = (adc_value >> 2) & 0x7F;

		// Перетворюємо отримане значення на символ
		char buffer[2];
		buffer[0] = (char)decoded_char; // Приводимо 8-бітове значення до символу
		buffer[1] = '\0'; // Завершення рядка
		if(decoded_char == '<'){
			LCD_Clear();
			continue;
		}

		// Очищуємо дисплей і виводимо символ
		LCD_String(buffer);

		_delay_ms(100); // Затримка перед наступним зчитуванням
	}
}
```