Katedra Nanometrologii

LABORATORIUM:
PRZETWORNIKI A/C i C/A

1. UKŁADY WEJŚCIA I WYJŚCIA PRZETWORNIKÓW A/C i C/A

IAD010402

wzn.pwr.edu.pl/materialy-dydaktyczne/przetworniki_ACiCA

ANDRZEJ SIKORA
JAKUB KONOPIŃSKI

IAD010402
V5 / 30.04.2024

1. **Zagadnienia do przygotowania na kartkówkę**
   a. Budowa i zasada działania interfejsów: równoległego, I²C, SPI, OneWire
   b. Podstawowe sposoby kodowania wartości w systemie dwójkowym oraz szesnastkowym
   c. Podstawowe parametry przetworników analogowo-cyfrowych i cyfrowo-analogowych
   d. Podstawowe układy pracy wzmacniacza operacyjnego
   e. Rodzaje architektur i sposoby działania przetworników analogowo-cyfrowych
   f. Zasady działania i komunikacji z przetwornikami badanymi w niniejszym ćwiczeniu
   g. Podstawowe informacje o architekturze układu Raspberry Pico

UWAGA! Przed przystąpieniem do laboratorium należy zapoznać się ze specyfiką działania i obsługi interfejsów w badanych układach, jak również zakresami podstawowych konfiguracji oraz trybów pracy.

2. **Układy wejściowe w przetwornikach analogowo - cyfrowych**

a) układy z wejściem asymetrycznym

Wejścia asymetryczne, nazywane także jednostronnymi, bazują na przesyle sygnału jednym kanałem (linią), dla którego poziomem odniesienia jest zazwyczaj masa. Wejścia asymetryczne są tanie i proste w implementacji, jednak ich wadą jest mała odporność na zakłócenia elektromagnetyczne, co może prowadzić do błędów w detekcji i interpretacji sygnałów.

b) układy z wejściem symetrycznym

Wejścia symetryczne (nazywane także dwustronnymi lub różnicowymi) funkcjonują w oparciu o przesył dwóch sygnałów przesuniętych fazowo względem siebie o 180 stopni (tzw. sygnały różnicowe). Zdekodowanie sygnału w układzie odbiorczym polega na wyznaczeniu różnicy między wartościami sygnałów. Wejścia symetryczne są bardziej odporne na zakłócenia elektromagnetyczne niż ich asymetryczne odpowiedniki, ponieważ wyindukowane w liniach sygnałowych zakłócenia mają ten sam znak. Dzięki temu łatwo jest je wyeliminować w procesie sumowania sygnałów różnicowych. Czyni je to bardziej niezawodnymi w zastosowaniach, w których występują zakłócenia. Ze względu na większą złożoność układową i wymaganą większą precyzję w odbiorze sygnałów, wyjścia i wejścia symetryczne są zazwyczaj droższe i trudniejsze w implementacji.

c) układy wielowejściowe multipleksowane

Układy wielowejściowe multipleksowane to układy elektroniczne, które umożliwiają przetwarzanie wielu sygnałów przy użyciu jednego układu scalonego. W takim układzie kilka lub kilkanaście sygnałów doprowadza się do poszczególnych wejść w układzie scalonym, gdzie są multipleksowane (przełączane) w określonych przedziałach czasowych i przetwarzane, a na wyjście danych przesyłane są pakiety informacji zawierające wartości chwilowe poszczególnych

sygnałów w momentach ich przetwarzania. W przypadku tego rozwiązania, jeśli na przeprowadzenie próbkowania sygnału potrzebny jest czas Δt, to na przeprowadzenie przetwarzania ośmiu sygnałów wymagany jest czas $8 \times \Delta t$. Niniejsze rozwiązanie jest skuteczne w przypadku gdy zachodzi konieczność próbkowania wielu sygnałów, a dynamika ich zmian jest znacząco mniejsza względem czasu przetwarzania, lub też wzajemna korelacja czasowa jest pomijalna.

d) układy wielowejściowe multipleksowane *sample/ hold*

W sytuacji analogicznej do omówionej powyżej, ale gdy dynamika ich zmian próbkowanych sygnałów jest porównywalna z czasem przetwarzania, a ich korelacja czasowa nie jest pomijalna (oznacza to, że zmiana wartości sygnału n+1-tego może ulec znaczącej zmianie podczas przetwarzania sygnału n-tego), może dojść do utraty istotnej informacji. W takim przypadku stosuje się przetworniki w których specjalne bloki funkcjonalne mogą w momencie t_0 zapamiętać chwilowe wartości sygnałów analogowych (*sample and hold*), a następnie pozwolić na dokonanie ich próbkowania w reżimie przetwornika multipleksowanego, jednak bez wpływu czasu niezbędnego na przeprowadzenie poszczególnych procesów przetwarzania. Niniejsze rozwiązanie jest efektywne w sytuacjach, gdzie sumaryczny czas przetwarzania wszystkich sygnałów nie przekracza dopuszczalnego interwału narzuconego przez szybkość zmiany sygnałów przetwarzanych.

e) układy wielowejściowe jednoczesnego przetwarzania

W przypadku gdy konieczne jest jednoczesne przetwarzanie kilku sygnałów analogowych, a wyżej wymienione rozwiązania są nieefektywne, stosuje się układu wielokanałowe, w których zaimplementowane są n-krotne struktury przetworników działających równolegle. Pozwala to uzyskać maksymalną wydajność przetwarzania, jednak ze względu na poziomo złożoności, jest rozwiązaniem najbardziej kosztownym.

f) przetworniki napięciowe

Najczęściej spotykanymi układami przetworników A/C są rozwiązania z wejściami napięciowymi, co oznacza że przetwarzaną wielkością elektryczną jest napięcie. Podczas lektury dokumentacji technicznej należy zwrócić uwagę zarówno na napięcie zasilania układu (może być rozdzielone napięcie bloku cyfrowego i analogowego), napięcie odniesienia (może być możliwość doprowadzenia zewnętrznego lub wykorzystania wewnętrznego) i wreszcie zakresu napięć poddawanych przetwarzaniu.

g) przetworniki prądowe

Znacznie mniej popularne są przetworniki prądowe, które mają wyspecjalizowane bloki przetwarzania, przez które wymusza się mierzony prąd. Oznacza to, że układ musi mieć wejście oraz wyjście. W praktyce sprowadza się to do zaimplementowania wewnątrz układu precyzyjnego rezystora oraz różnicowego układu mierzącego spadek napięcia na tymże rezystorze.

h) przetworniki specjalnego zastosowania

Do grupy przetworników specjalnego zastosowania zaliczyć można układy dedykowane realizacji zadań próbkowania specyficznych sygnałów – m.in. kodeki audio, mierniki energii elektrycznej, przetworniki do ekranów dotykowych rezystancyjne/ pojemnościowe, izolowane, przetworniki sygnałów ultradźwiękowych, sygnału wizyjnego. Ich struktury są wyspecjalizowane zarówno w kontekście parametrów przetwarzania (rozdzielczość oraz częstotliwość próbkowania, wbudowane odpowiednie filtry oraz układy wzmacniające i wreszcie algorytmy przetwarzania danych), jak również kodowania i przesyłania informacji do urządzenia nadrzędnego (np. dedykowane protokoły przesyłu danych audio z wykorzystaniem interfejsu I^2S).

i) przetworniki wielkości nieelektrycznych

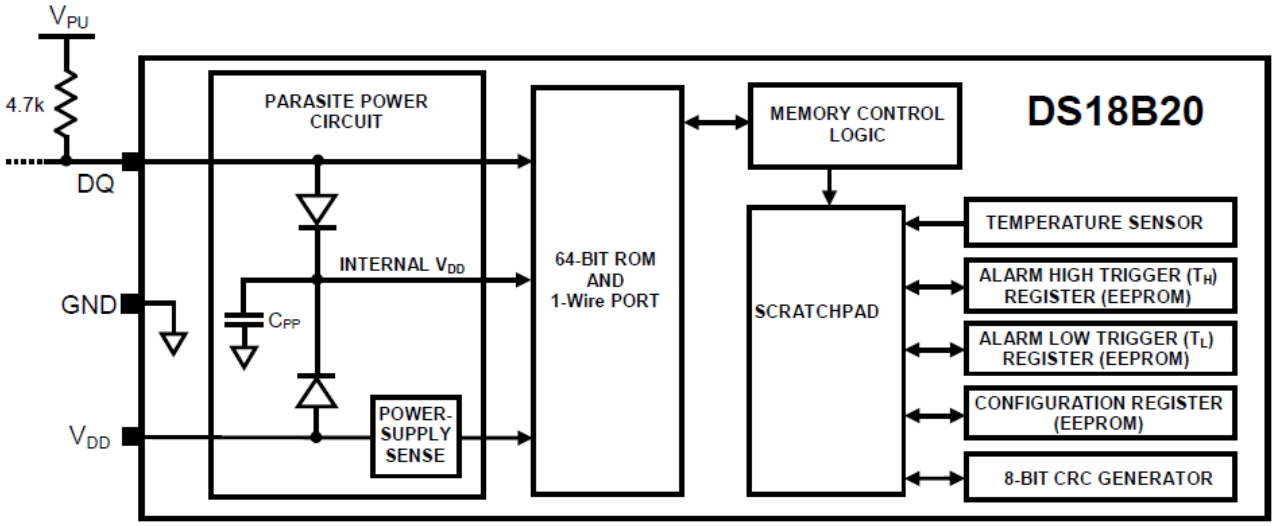
Omawiane tu przetworniki wielkości nieelektrycznych to urządzenia elektroniczne, stanowiące połączenie dwóch bloków funkcjonalnych: przetwornika wielkości nieelektrycznej na elektryczną dokonującego przetwarzania sygnałów z różnych rodzajów czujników, mierzących wartości nieelektryczne, takie jak temperatura, wilgotność, ciśnienie, ruch, dźwięk, światło, wibracje i wiele innych, oraz przetwornika analogowo-cyfrowego, pozwalającego na przesłanie zmierzonej wielkości w formie cyfrowej do urządzenia nadrzędnego.

3. Układy przetworników dostępne na makiecie dydaktycznej

a) DS18B20 – interfejs: 1-Wire

Kluczowe parametry przetwornika:

- Interfejs 1-Wire® wymagający jednej linii do komunikacji
- Każde urządzenie ma unikatowy 64-bitowy kod identyfikacyjny zapisany w wewnętrznej pamięci ROM
- Nie wymaga stosowania dodatkowych podzespołów
- Może być zasilany przez linię danych, w zakresie napięć od 3.0V do 5.5V
- Rozwiązanie pozwalające na łatwą implementację wielopunktowego pomiaru temperatury
- Zakres przetwarzania od -55°C do +125°C (-67°F do +257°F)
- Dokładność ±0.5°C w przedziale temperatur od -10°C do +85°C
- Rozdzielczość przetwarzania konfigurowalna w zakresie od 9 do 12 bitów
- Maksymalny czas przetwarzania 750ms (przy rozdzielczości 12 bitów)



Rys. 1. Schemat blokowy przedstawiający budowę układu DS18B20

Adres urządzenia zakodowany jest w sposób przedstawiony na poniżej grafice. Prawidłowe zidentyfikowanie numeru seryjnego układu warunkuje uruchomienie procesu przetwarzania i uzyskanie jego wyniku.

8-BIT CRC		48-BIT SERIAL NUMBER		8-BIT FAMILY CODE (28h)	
MSB	LSB	MSB	LSB	MSB	LSB

Rys. 2. Struktura numeru seryjnego układu (ROM)

Wynik przetwarzania zakodowany jest w dwóch słowach 8-bitowych zgodnie z poniżej przedstawionym układem.

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
LS BYTE	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}
	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
MS BYTE	S	S	S	S	S	2^6	2^5	2^4

S = SIGN

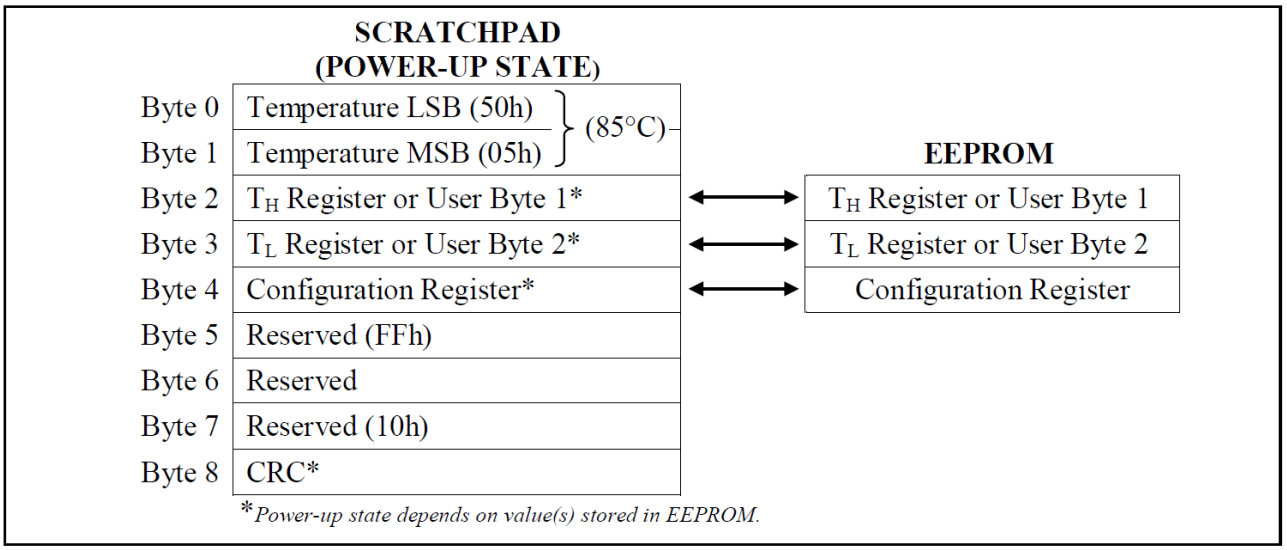
Rys. 3. Struktura wyniku przetwarzania

Poniższa tabela przedstawia przykładowe wartości temperatur i odpowiadające im wartości słów w kodzie binarnym oraz szesnastkowym.

Tab. 1 Przykładowe sposoby kodowania wyników przetwarzania

TEMPERATURE(°C)	DIGITAL OUTPUT (BINARY)	DIGITAL OUTPUT (HEX)
+125	0000 0111 1101 0000	07D0h
+85*	0000 0101 0101 0000	0550h
+25.0625	0000 0001 1001 0001	0191h
+10.125	0000 0000 1010 0010	00A2h
+0.5	0000 0000 0000 1000	0008h
0	0000 0000 0000 0000	0000h
-0.5	1111 1111 1111 1000	FFF8h
-10.125	1111 1111 0101 1110	FF5Eh
-25.0625	1111 1110 0110 1111	FE6Fh
-55	1111 1100 1001 0000	FC90h

Wyniki pomiarów oraz dane konfiguracyjne przetwornika zapisane są w jego pamięci wewnętrznej. Opis poszczególnych rejestrów i ich zawartość przedstawiono na poniższej ilustracji.



Rys. 4. Opis rejestrów przetwornika

Dane mogą być pobierane i przesyłane do rejestrów w porządku przedstawionym na poniżej ilustracji.

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
S	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Rys. 5. Opis porządku w 8-bitowym słowie danych

Procedura nawiązania komunikacji z układem DS18B20 przebiega następująco:

Krok 1. Inicjalizacja

Krok 2. Wywołanie adresu ROM (wraz z następującą wymianą danych)

Krok 3. Komenda funkcji DS18B20 (wraz z następującą wymianą danych)

Kluczowe jest, aby niniejsza sekwencja była realizowana przy każdej komunikacji z układem DS18B20, ponieważ w przypadku pominięcia któregoś z opisanych powyżej etapów, przetwornik nie będzie reagował. Do wyjątków od tej reguły należą: procedura skanowania adresu ROM [F0h] oraz skanowanie alarmów [ECh]. Po zrealizowaniu jednego z tych poleceń ROM, układ nadrzędny musi wznowić komunikację począwszy od kroku 1.

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
0	R1	R0	1	1	1	1	1

Rys. 6. Struktura rejestru konfiguracyjnego

Układ umożliwia przetwarzanie sygnałów ze zdefiniowaną przez użytkownika rozdzielczością. Zestawienie zasad konfiguracji tego parametru (wartości wpisywane w rejestrze konfiguracyjnym) przedstawiono w tabeli 2.

Lokalizacja rejestru konfiguracyjnego została przedstawiona na rysunku 4.

Tab 2. Konfigurowanie rozdzielczości pomiarowej przetwornika

R1	R0	RESOLUTION (BITS)	MAX CONVERSION TIME	
0	0	9	93.75 ms	(t_{CONV}/8)
0	1	10	187.5 ms	(t_{CONV}/4)
1	0	11	375 ms	(t_{CONV}/2)
1	1	12	750 ms	(t_{CONV})

Zestawienie komend uruchamiających poszczególne działania przedstawiono w tabeli 3.

Tab. 3. Zestawienie komend obsługujących komunikację z układem DS18B20.

COMMAND	DESCRIPTION	PROTOCOL	1-Wire BUS ACTIVITY AFTER COMMAND IS ISSUED	NOTES
TEMPERATURE CONVERSION COMMANDS				
Convert T	Initiates temperature conversion.	44h	DS18B20 transmits conversion status to master (not applicable for parasite-powered DS18B20s).	1
MEMORY COMMANDS				
Read Scratchpad	Read the entire scratchpad including the CRC byte.	BEh	DS18B20 transmits up to 9 data bytes to master.	2
Write Scratchpad	Writes data into scratchpad bytes 2, 3, and 4 (T_H, T_L, and configuration registers).	4Eh	Master transmits 3 data bytes to DS18B20.	3
Copy Scratchpad	Copies T_H, T_L, and configuration register data from scratchpad to EEPROM	48h	None	1
Recall E^2	Recalls T_H, T_L, and configuration register data from EEPROM to scratchpad	B8h	DS18B20 transmits recall status to master	
Read Power Supply	Signals DS18B20 power supply mode to the master.	B4h	DS18B20 transmits supply status to master.	

Opis sekwencji komunikacyjnej z układem DS18B20 przedstawiono w tabeli 4.

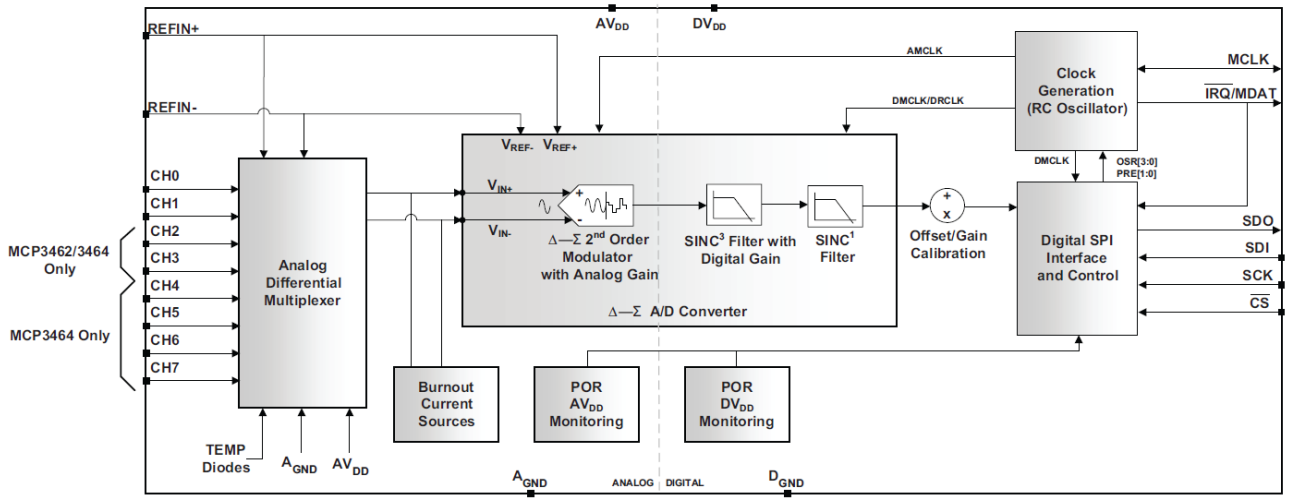
Tab. 4. Zestawienie poszczególnych elementów tworzących ramkę komunikacji z układem DS18B20.

MASTER MODE	DATA (LSB FIRST)	COMMENTS
T_x	Preset	Master issues reset pulse.
R_x	Presence	DS18B20s respond with presence pulse.
T_x	55h	Master issues Match ROM command.
T_x	64-bit ROM code	Master sends DS18B20 ROM code.
T_x	44h	Master issues Convert T command.
T_x	DQ line held high by strong pullup	Master applies strong pullup DQ for the durstion of the conversion (t_{CONV}).
T_x	Reset	Master issues reset pulse.
R_x	Presence	DS18B20s respond with reset pulse.
T_x	55h	Master issues Match ROM command.
T_x	64-bit ROM code	Master sends DS18B20 ROM code.
T_x	BEh	Master issues Read Scratchpad command.
R_x	9 data bytes	Master reads entire scratchpad including CRC. The master then recalculates the CRC of the first eigth data bytes from the scratchpad and compares the calculated CRC with read CRC (byte 9). If they match, the master continues; if not, the read operation is repeated.

b) MCP3461, protokół: SPI

Kluczowe parametry przetwornika:
• Jedno/ dwu/ cztero-kanałowe wejście różnicowe lub dwu/ cztero/ ośmio kkanałowe wejście asymetryczne
• Rozdzielczość przetwarzania 16-bitów
• Programowalna szybkość przetwarzania, do 153.6 ksps
• Programowalne wzmocnienie: od 0.33x do 64x
• 97.2 dB SINAD, -116 dBc THD, 120 dBc SFDR
(Gain = 1x, 4800 SPS)
• Niski dryf temperaturowy:
- Offset wynikający z dryfu: 4/Gain nV/°C (AZ_MUX = 1)
- Błąd wzmocnienia wynikający z dryfu: 0.5 ppm/°C (Gain = 1x)
• Niski poziom szumów: 3.2 µVRMS (Gain = 16x, 9600 SPS)
• Szeroki zakres przetwarzania napięcia: od 0V do AVDD
• Wewnętrzne źródło taktowania z możliwością podłączenia zewnętrznego zegara
• Bardzo niski pobór prądu w trybie wyłączenia (< 5 µA)
• Wewnętrzny czujnik temperatury
• 16-Bit Digital Offset and Gain Error Calibration Registers
• Wewnętrzny sekwencer przetwarzania (SCAN Mode) wspierający automatyczne multipleksowane
• Dedykowany pin przerwań ułatwiający przesył danych
• Zaawansowane funkcjonalności bezpieczeństwa:
- 16-bitowa CRC dla bezpiecznego przesyłania interfejsem SPI
- 16-bitowa CRC oraz IRQ dla bezpiecznej konfiguracji
- Rejestry z dostępem zabezpieczonym słowem 8-bitowym
- Rejestry kontrolno-monitorujące wspomagające diagnostykę układu
• Interfejs SPI o szybkości taktowania 20 MHz pracujący w Mode 0,0 oraz 1,1
• AVDD: 2.7V-3.6V
• DVDD: 1.8V-3.6V
• Rozszerzony zakres temperatury pracy: -40°C to +125°C

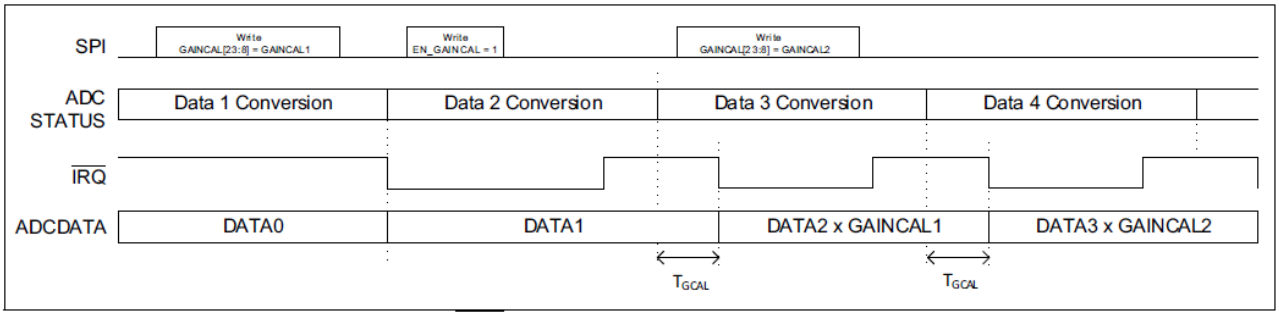
Strukturę układu i jego podstawowe bloki funkcjonalne przedstawiono na rysunku 7. Uruchomienie przetwarzania tym układem wymaga w pierwszej kolejności przeprowadzenia konfiguracji (proszę sprawdzić notę katalogową, tabela 6). Należy zwrócić uwagę na fakt, że do układu nie doprowadzono zewnętrznego sygnału zegarowego. Uruchomienie przetwarzania wyzwalane jest po wysłaniu odpowiedniego polecenia.



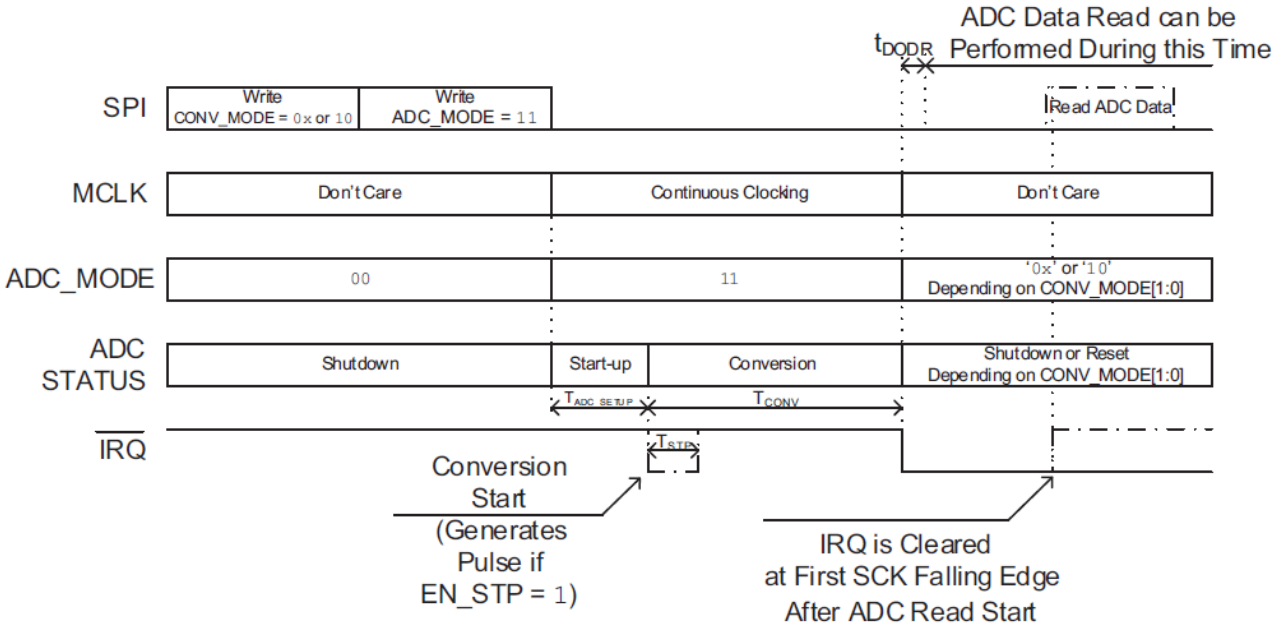
Rys. 7. Struktura i kluczowe bloki funkcjonalne układu MCP3464

Tab. 5. Format danych – sposób kodowania DATA_FORMAT[1:0] = 1x (17-BIT CODING)

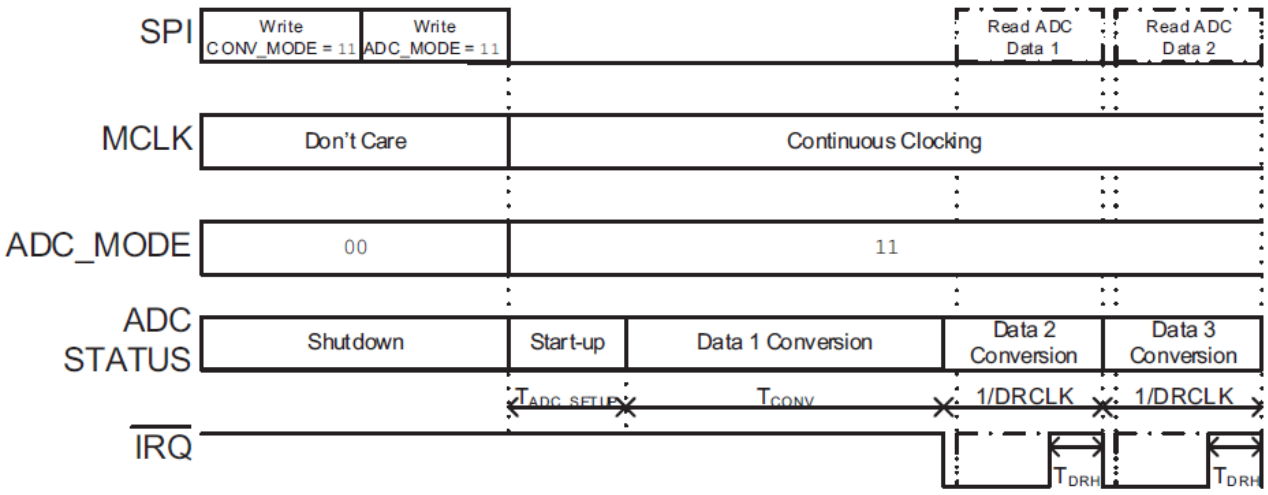
Equivalent Input Volatge	ADC Output Code (SGN + DATA[15:0])	Hexadecimal	Decimal
> 2 V_{REF} – 1 LSb	0111 1111 1111 1111 1	0x0FFFF	+65535
2 V_{REF} – 2 LSb	0111 1111 1111 1111 0	0x0FFFE	+65534
V_{REF} + 1 LSb	0100 0000 0000 0000 1	0x08001	+32769
V_{REF}	0100 0000 0000 0000 0	0x08000	+32768
V_{REF} – 1 LSb	0011 1111 1111 1111 1	0x07FFF	+32767
V_{REF} – 2 LSb	0011 1111 1111 1111 0	0x0FFE	+32766
1 LSb	0000 0000 0000 0000 1	0x00001	+1
0	0000 0000 0000 0000 0	0x00000	0
- 1 LSb	1111 1111 1111 1111 1	0x1FFFF	- 1
- V_{REF} + 1 LSb	1100 0000 0000 0000 1	0x18001	-32767
- V_{REF}	1100 0000 0000 0000 0	0x18000	-32768
- V_{REF} - 1 LSb	1011 1111 1111 1111 1	0x17FFF	-327689
-2 V_{REF} - 1 LSb	1000 0000 0000 0000 1	0x10001	-65535
< -2 V_{REF}	1000 0000 0000 0000 0	0x10000	-65536



Rys. 8. Struktura transmisji danych w układzie MCP3464



Rys. 9. Struktura transmisji danych w układzie MCP3464 – znaczenie sygnału IRQ



Rys. 10. Struktura transmisji danych w układzie MCP3464 – konwersja wielokanałowa

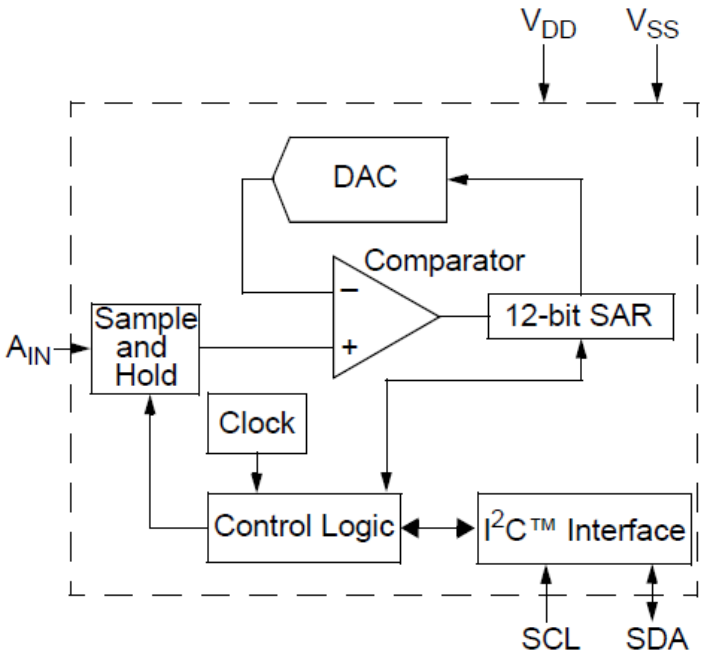
Tab 6. Zestawienie komend sterujących układem MCP3464

CMD[5:2]	CMD[1:0]	Command Description
0xxx	00	Don't Care
100x	00	Don't Care
1010	00	ADC Conversion Start/Restart Fast Command (Overwrites ADC_MODE[1:0] = 10)
1011	00	ADC Standby Mode Fast Command (Overwrites ADC_MODE[1:0] = 10)
1100	00	ADC Shutdown Mode Fast Command (Overwrites ADC_MODE[1:0] = 00)
1101	00	Full Shutdown Mode Fast Command (Overwrites CONFIG0[7:0] = 0x00)
1110	00	Device Full Reset Fast Command (Resets Whole Register Map to Default Value)
1111	00	Don't Care
ADDR	01	Static Read of Register Address, ADDR
ADDR	10	Incremental Write Starting at Register Address, ADDR
ADDR	11	Incremental Read Starting at Register Address, ADDR

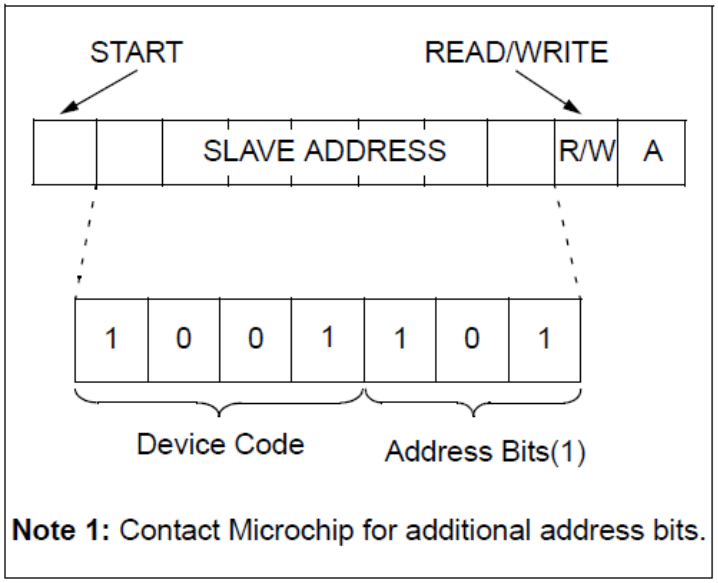
c) I²C
MCP3221

• rozdzielczość przetwarzania 12-bitów
• ±1 LSB DNL, ±2 LSB INL max.
• maksymalne obciążenie prądowe podczas przetwarzania 250 µA
• typowy prąd w trybie spoczynku 5 nA, maksymalny prąd w trybie spoczynku 1 µA max.
• protokół komunikacji I2C™
- szybkość transmisji 100 kHz I2C Standard Mode
- szybkość transmisji 400 kHz I2C Fast Mode
• możliwość obsługi do 8 układów na jednej magistrali 2-przewodowej
• szybkość próbkowania 22.3 ksps w trybie I2C Fast Mode
• wejścia analogowe asymetryczne
• wbudowana struktura *sample and hold*
• wbudowany zegar taktujący przetwarzanie
• zasilanie unipolarne w zakresie napięć: 2.7V to 5.5V

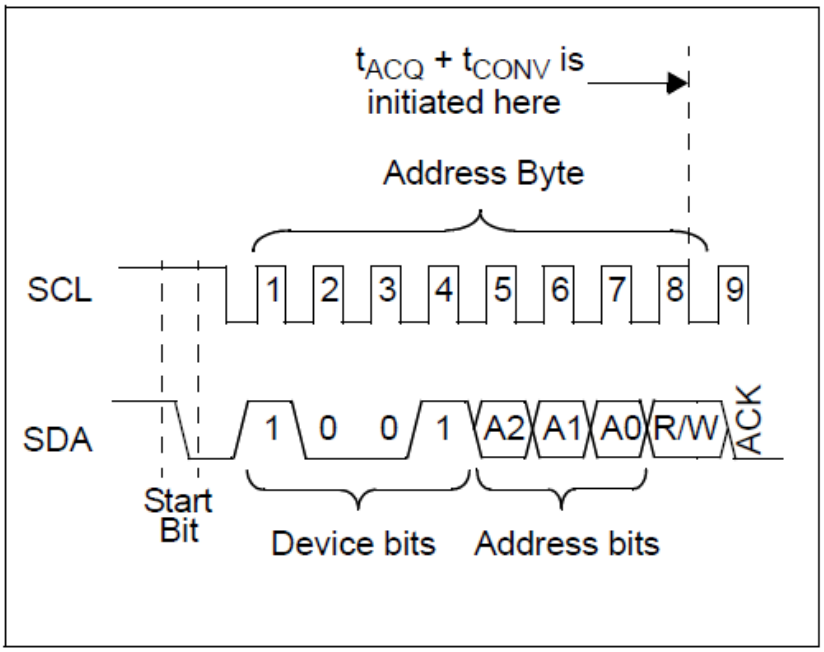
• zakres temperaturowy pracy
- wersja przemysłowa: -40°C to +85°C
- wersja rozszerzona: -40°C to +125°C



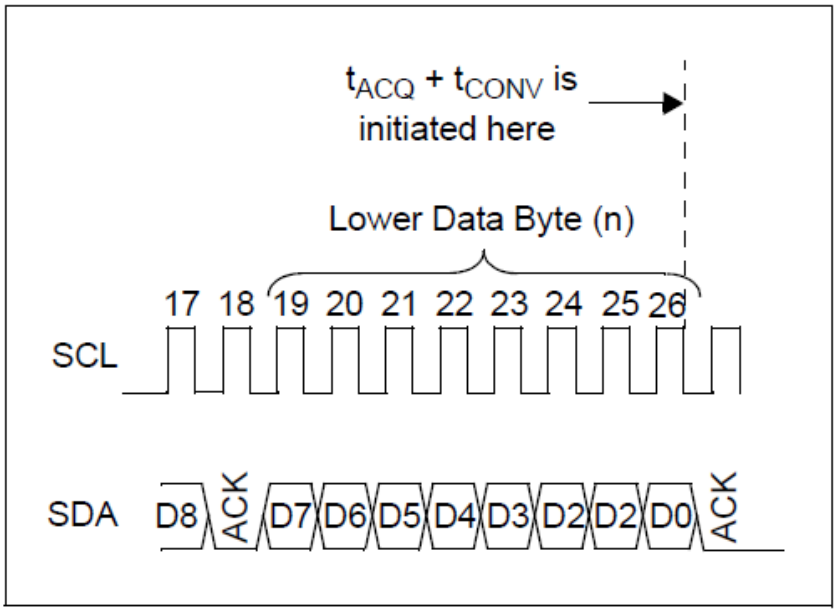
Rys. 11. Struktura i kluczowe elementy układu MCP3221



Rys. 12. Struktura bajtu wywołania w komunikacji z układem MCP3221



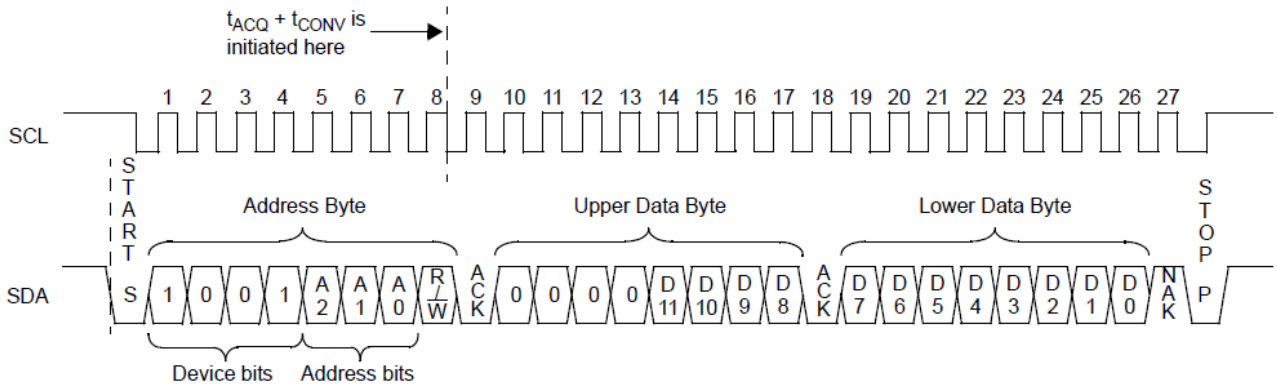
Rys. 13. Struktura bajtu wywołania w komunikacji z układem MCP3221



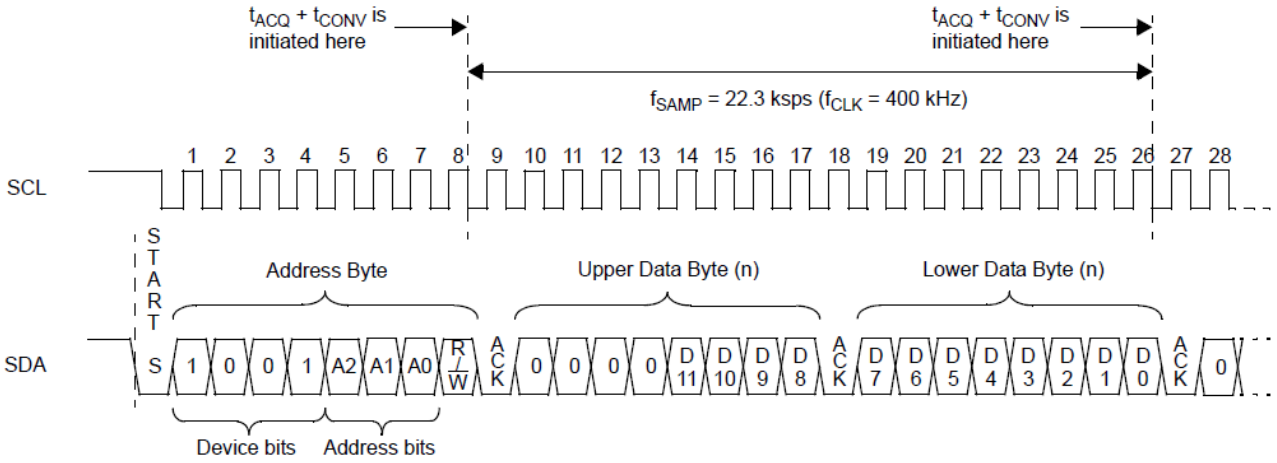
Rys. 14. Inicjalizacja konwersji, przetwarzanie ciągłe

Tab 7. Adresy urządzeń (bity A0-A2) w zależności od wersji

Wersja układu	Wersja adresu	Opis na obudowie
MCP3221A0T-I/OT	000	EE
MCP3221A1T-I/OT	001	EH
MCP3221A2T-I/OT	010	EB
MCP3221A3T-I/OT	011	EC
MCP3221A4T-I/OT	100	ED
MCP3221A5T-I/OT	101	S1*
MCP3221A6T-I/OT	110	EF
MCP3221A7T-I/OT	111	EG
MCP3221A0T-E/OT	000	GE
MCP3221A1T-E/OT	001	GH
MCP3221A2T-E/OT	010	GB
MCP3221A3T-E/OT	011	GC
MCP3221A4T-E/OT	100	GD
MCP3221A5T-E/OT	101	GA*
MCP3221A6T-E/OT	110	GF
MCP3221A7T-E/OT	111	GG



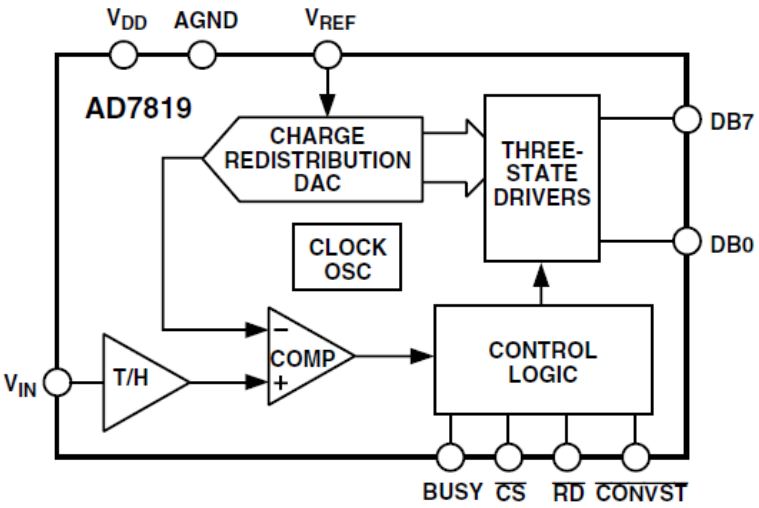
Rys. 14. Diagram przedstawiający transmisję danych przetwarzania pojedynczego



Rys. 15. Diagram przedstawiający transmisję danych przetwarzania ciągłego

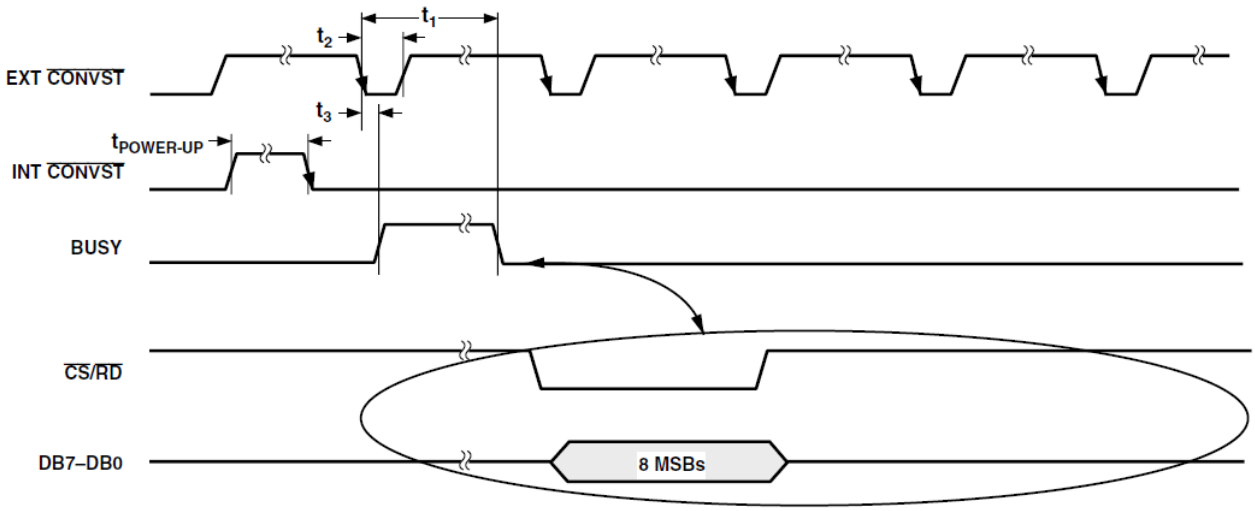
d) Układ z magistralą równoległą
AD7819

- przetwornik 8-bitowy
- wbudowany blok *Track and Hold*
- zakres napięć roboczych: 2.7 V to 5.5 V
- interfejs równoległy 8-bitowy
- pobór mocy 10.5 mW, VDD = 3 V
- automatyczne przechodzenie w stan uśpienia
- zakres napięć wejściowych: 0 V to VREF
- zakres napięcia odniesienia: 1.2 V to VDD

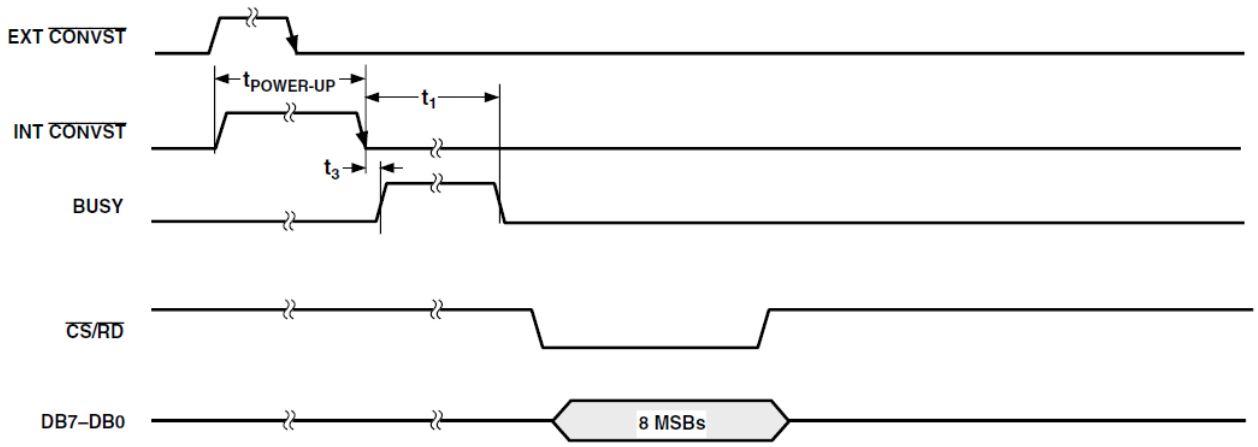


Rys. 16. Struktura i kluczowe elementy układu AD7819

Sposób obsługi przetwornika przedstawiono na rysunkach 17 i 18.



Rys. 17. Diagram komunikacji z układem w trybie 1 (High Speed Sampling)

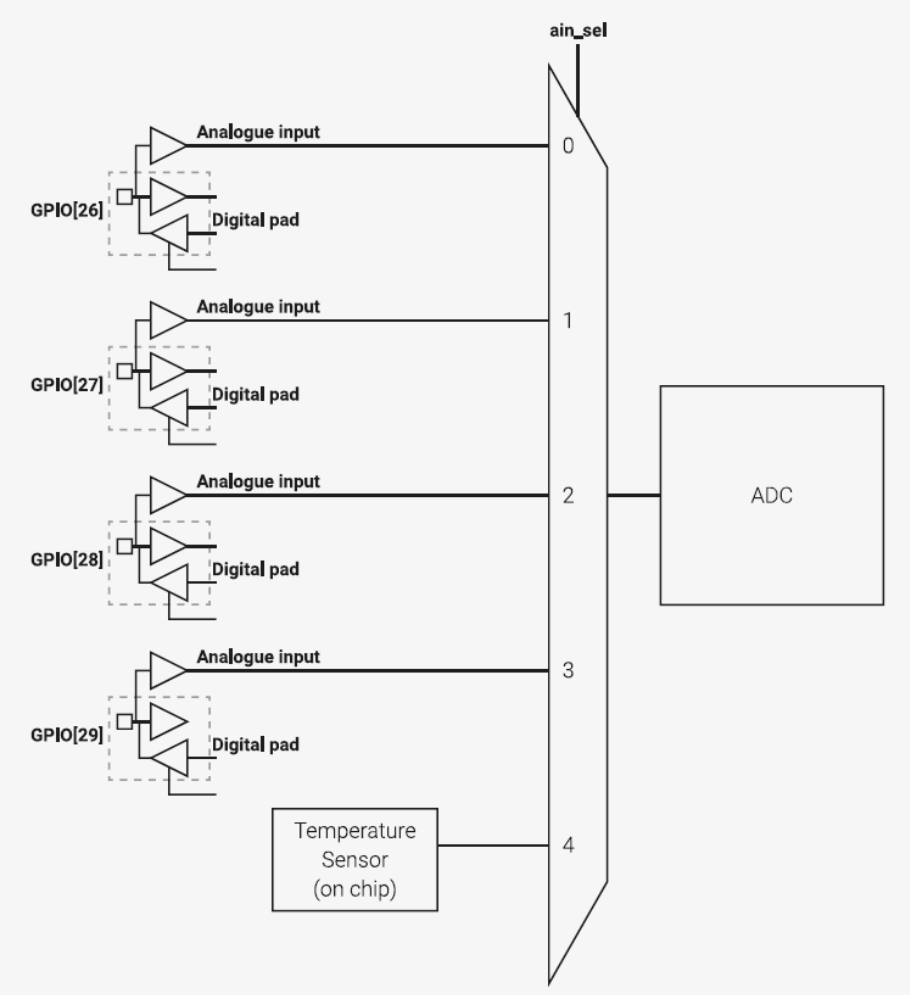


Rys. 18. Diagram komunikacji z układem w trybie 2 (Automatic Power Down)

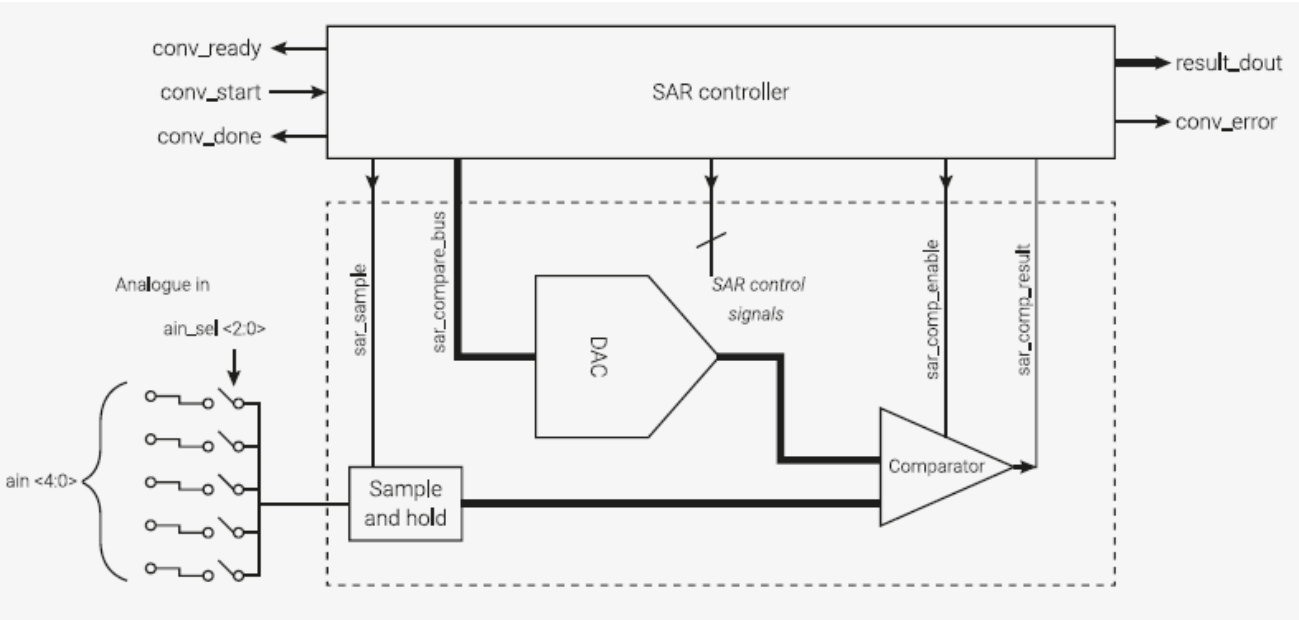
W układzie AD7819 aktywacja przetwarzania wywoływana jest poprzez stan wysoki w linii CONVST. Linia BUSY informuje o trwającym przetwarzaniu, natomiast linia CS/RD pozwala na aktywację wyjść i odczytanie wyniku przetwarzania (dokumentacja układu).

e) Przetwornik wewnętrzny
RP2040 (Raspberry Pico)

- przetwornik typu SAR
- szybkość próbkowania 500 kS/s (przy zastosowaniu niezależnego zegara 48MHz)
- przetwornik 12-bitowy
- Pięć multipleksowanych wejść:
  - cztery wejścia dostępne w obudowie, multipleksowane z GPIO[29:26]
  - jedno wejście obsługujące wewnętrzny czujnik temperatury
- obsługa FIFO
- obsługa przerwań
- obsługa DMA



Rys. 19. Struktura stopnia wejściowego przetwornika



Rys. 20. Struktura przetwornika

Tab. 8. Adresy rejestrów konfiguracyjnych przetwornika

Offset	**Name**	**Info**
0x00	CS	ADC Control and Status
0x04	RESULT	Result of most recent ADC conversion
0x08	FCS	FIFO control and status
0x0c	FIFO	Conversion result FIFO
0x10	DIV	Clock divider. If non-zero, CS_START_MANY will start conversions at regular intervals rather then back-to-back. The divider is reset when either of these fields are written. Total period is ! + INT + FRAC/256
0x14	INTR	Raw Interrupts
0x18	INTE	Interrupt Enable
0x1c	INTF	Interrupt Force
0x20	INTS	Interrupt status after masking & forcing

Tab. 9. Struktura rejestru obsługi przetwarzania

Bits	Name	Description	Type	Reset
20:16	RROBIN	Round-robin sampling. 1 bit per channel. Set all bits to 0 to disable. Otherwise, the ADC will cycle through each enabled channel in round-robin fashion. The first channel to be sampled will be the one currently indicated by ANISEL. ANISEL will be updated after each conversion with the newly-selected channel.	RW	0x00
15	Reserved.	-	-	-
14:12	ANISEL	Select analog mux input. Updated automatically in round-robin mode.	RW	0x0
11	Reserved.	-	-	-

10	ERR_STICKY	Some past ADC conversion encountered an error. Write 1 to clear.	WC	0x0
9	ERR	The most recent ADC conversion encountered an error, result is undefined or noisy	RO	0x0
8	READY	1 if the ADC is ready to start a new conversion. Implies any previous conversation has completed. 0 whilst conversation in progress.	RO	0x0
7:4	Reserved.	-	-	-
3	START_MANY	Continuously perform conversation whilst this bit is 1. A new conversion will start immediately after the previous finishes.	RW	0x0
2	SSTART_ONCE	Start a single conversation. Self-clearing. Ignored if start_many is asserted	SC	0x0
1	TS_EN	Power on temperature sensor. 1- enabled. 0 – disabled.	RW	0x0
0	EN	Power on ADC and enable its clock. 1 – enabled. 0 – disabled.	RW	0x0

Tab. 10. Struktura rejestru wyniku przetwarzania

Bits	**Name**	**Description**	**Type**	**Reset**
31:12	Reserved.	-	-	-
11:0	NONAME		RO	0x000

Przetwarzanie pojedyncze polega na zmianie stanu bitu CS.START_ONCE na 1. Uruchomienie konwersji zasygnalizowanie będzie zmianą stanu bitu CS.READY na poziom niski. po 96 syklach zegara of clk_adc, bit CS.READY przyjmie stan wysoki. Wynik konwersji 12-bitowej dostępny będzie w rejestrze RESULT.

Wybór kanału przetwornika ADC polega na zapisaniu odpowiedniej kombinacji CS.AINSEL, przed rozpoczęciem przetwarzania. Wartości AINSEL w zakresie 0…3 pozwolą na aktywację ADC wejść na liniach GPIO 26…29. Wartość AINSEL równa 4, spowoduje wybór czujnika temperatury.

4. Opis budowy i sposobu działania makiety dydaktycznej

Kluczowymi elementami makiety jest analogowy tor kondycjonowania sygnałów, przetworniki ADC oraz jednostka nadrzędna pozwalająca kontrolować przetworniki, jaką jest Raspberry Pico.

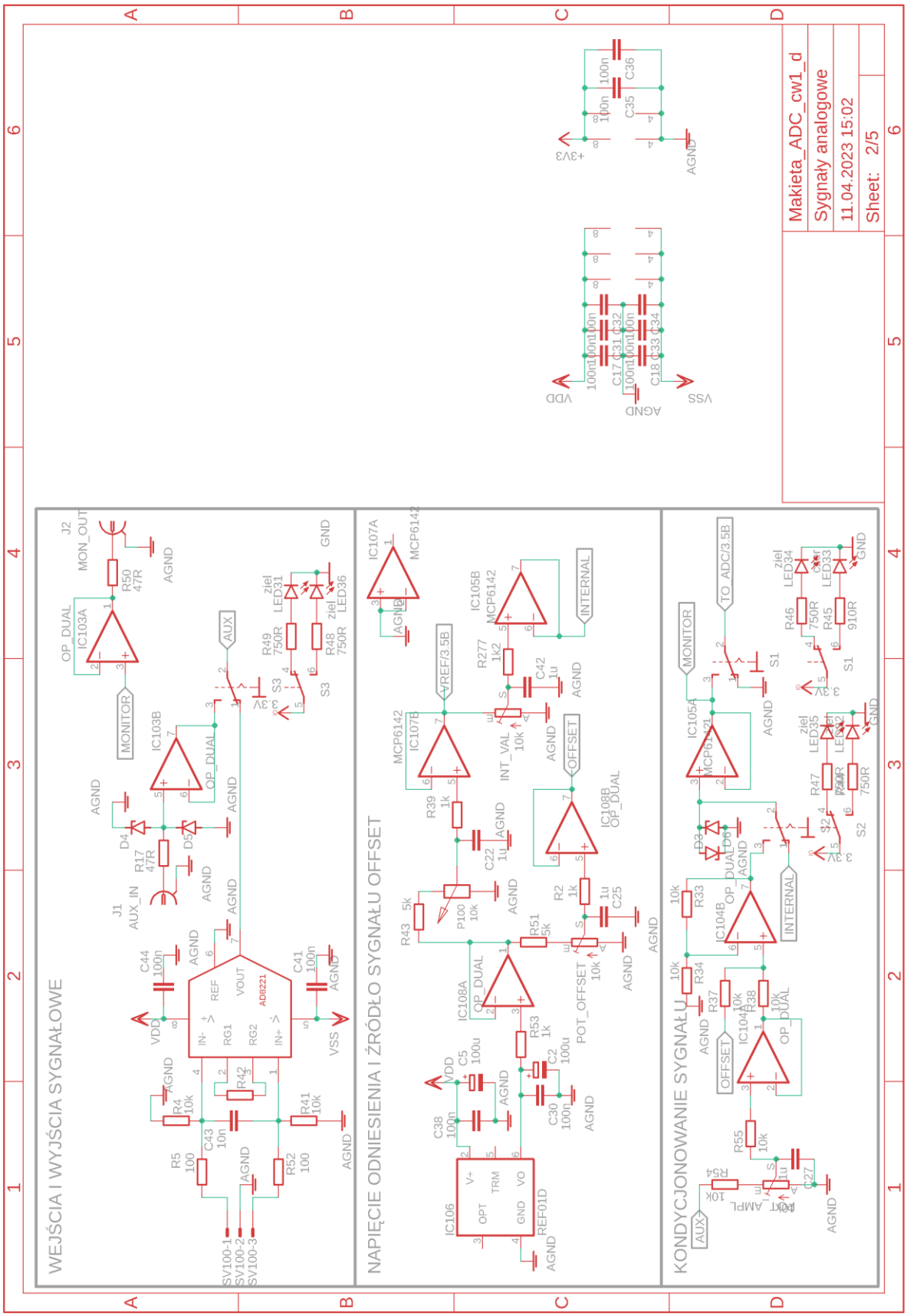
Na rysunkach 21-23 przedstawiono schematy tychże bloków, na rysunku 24 – rozkład elementów na płytce drukowanej.

Blok kondycjonowania (rys. 21) pozwala na przygotowanie sygnału do wprowadzenia do przetworników, w szczególności dopasowania zakresu zmian napięcia wejściowego (AMPLITUDE), oraz poziomów tych zmian (OFFSET). Ponadto, układ ma możliwość dokonywania nastaw wartości napięcia stałego generowanego na płytce przy użyciu trzeciego potencjometru (INTERNAL). Blok ten pozwala ponadto dokonywać wyboru źródła sygnału przy użyciu dwóch przełączników (S2 i S3), gdzie do dyspozycji jest wejście asymetryczne oraz symetryczne, a przy użyciu trzeciego przełącznika (S1) załączyć doprowadzenie sygnału próbkowanego do wejść przetworników. Przełączenie przełącznika S1 w pozycję doprowadzającą sygnał przetwarzany do wejść przetworników jest możliwe dopiero po upewnieniu się, że sygnał ten mieści się w zakresie przetwarzania (nie będzie wykraczał poza podany w karcie katalogowej przetwornika przedział, co ma chronić przetwornik przed uszkodzeniem).

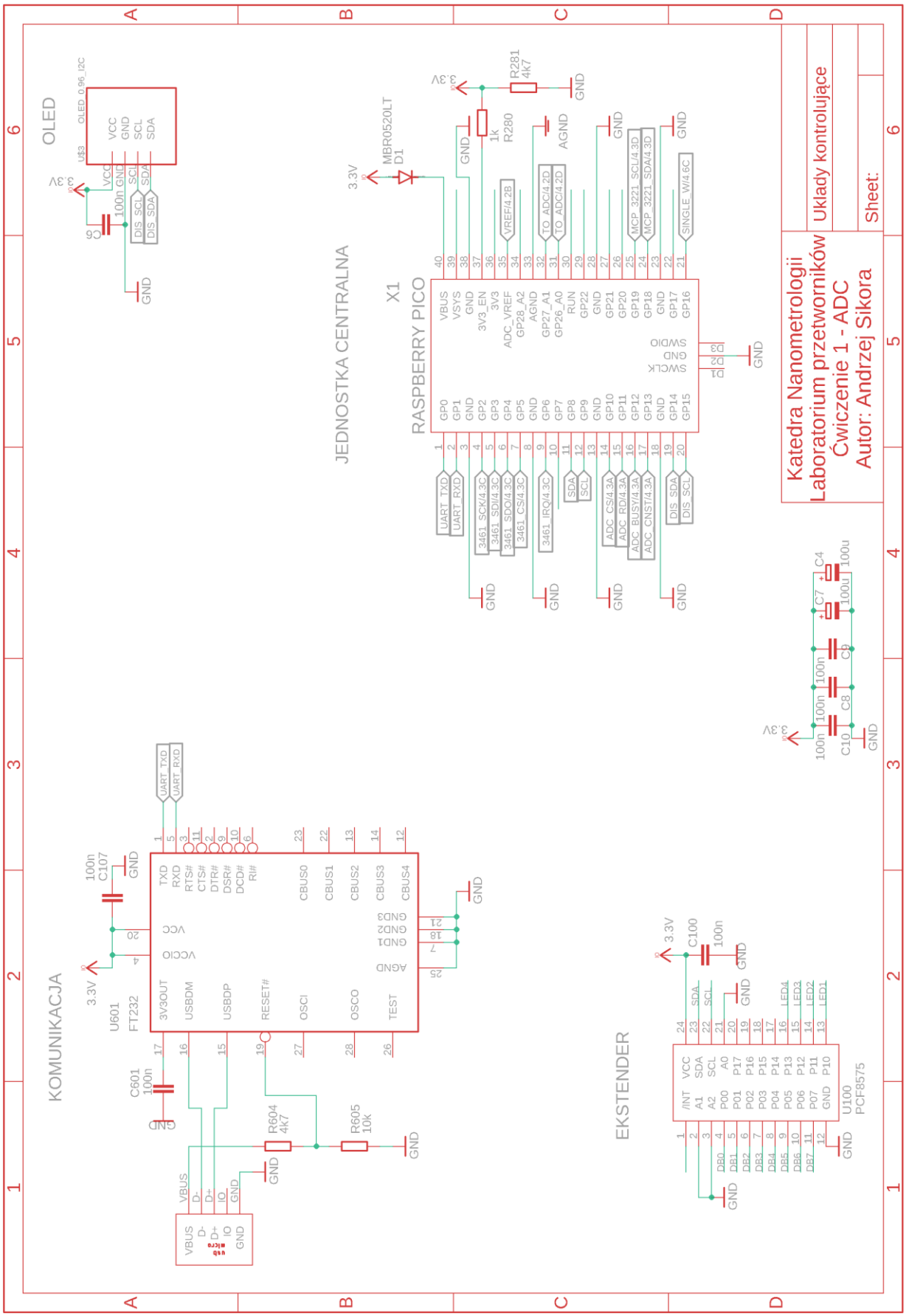
W bloku przetworników (rys. 22) zaimplementowano układy opisane w rozdziale 2 niniejszego dokumentu. Linie transmisji danych są doprowadzone do jednostki nadrzędnej jakim jest Raspberry Pico. Rozwiązanie to pozwala na obsługę poszczególnych interfejsów (One Wire, I²C, SPI, równoległy) zgodnie z ich specyfiką, przy wykorzystaniu poleceń w języku MicroPython. Uproszczony widok połączeń, ułatwiający konfigurację połączeń, przedstawiony jest na rysunku 25.

Blok jednostki nadrzędnej (rys. 23) zawiera Raspberry Pico wraz z układem konwersji I2C – port równoległy (PCF8575), pozwalającym na zwiększenie możliwości komunikacyjnych z peryferiami obecnymi na płytce. Ponadto, w układzie zaimplementowany jest moduł konwersji sygnału UART-USB, który może być wykorzystany w procesie wysokoprzepustowej transmisji danych próbkowania o wyższych szybkościach próbkowania.

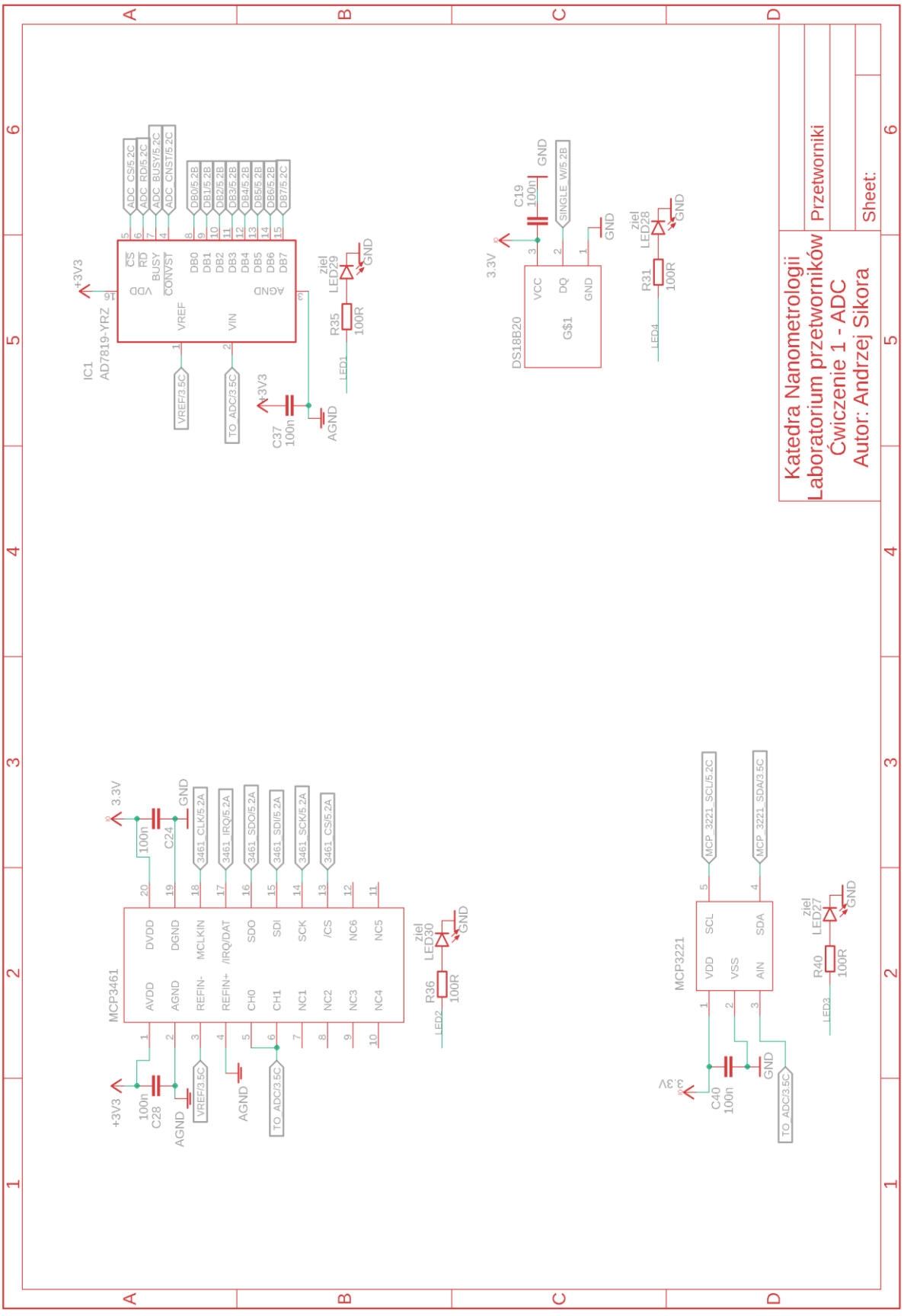
Widok płytki drukowanej przedstawiony na rysunku 24 pozwala zauważyć opisy wskazujące na drogę sygnałów: od wejść / wewnętrznego zadajnika napięcia (INTERNAL), przez blok kondycjonowania sygnału (AMPLITUTE/ OFFSET), przełącznik S1 (KONWERSJA) do poszczególnych przetworników.



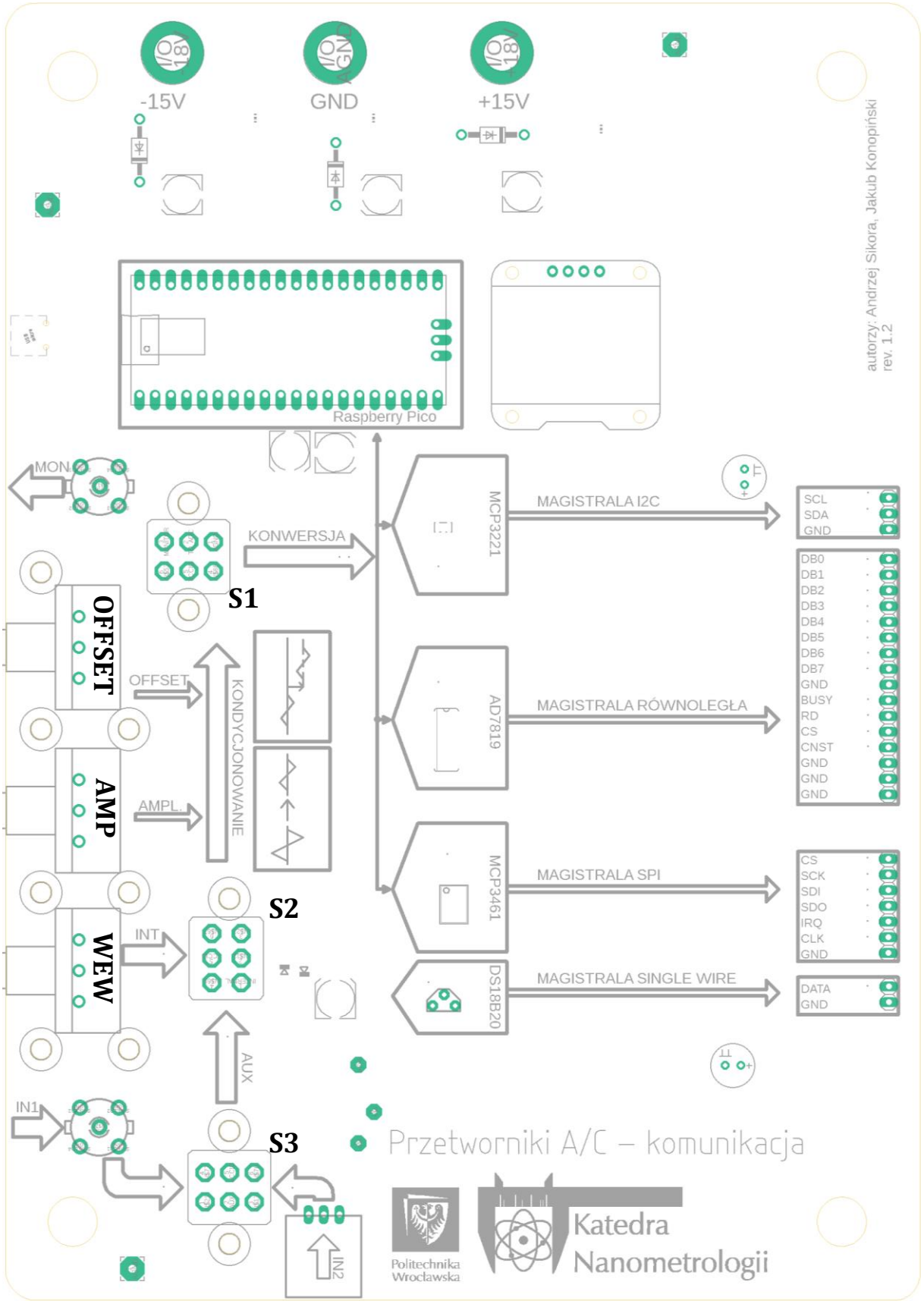
Rys. 21. Schematy bloku analogowych torów kondycjonowania sygnałów



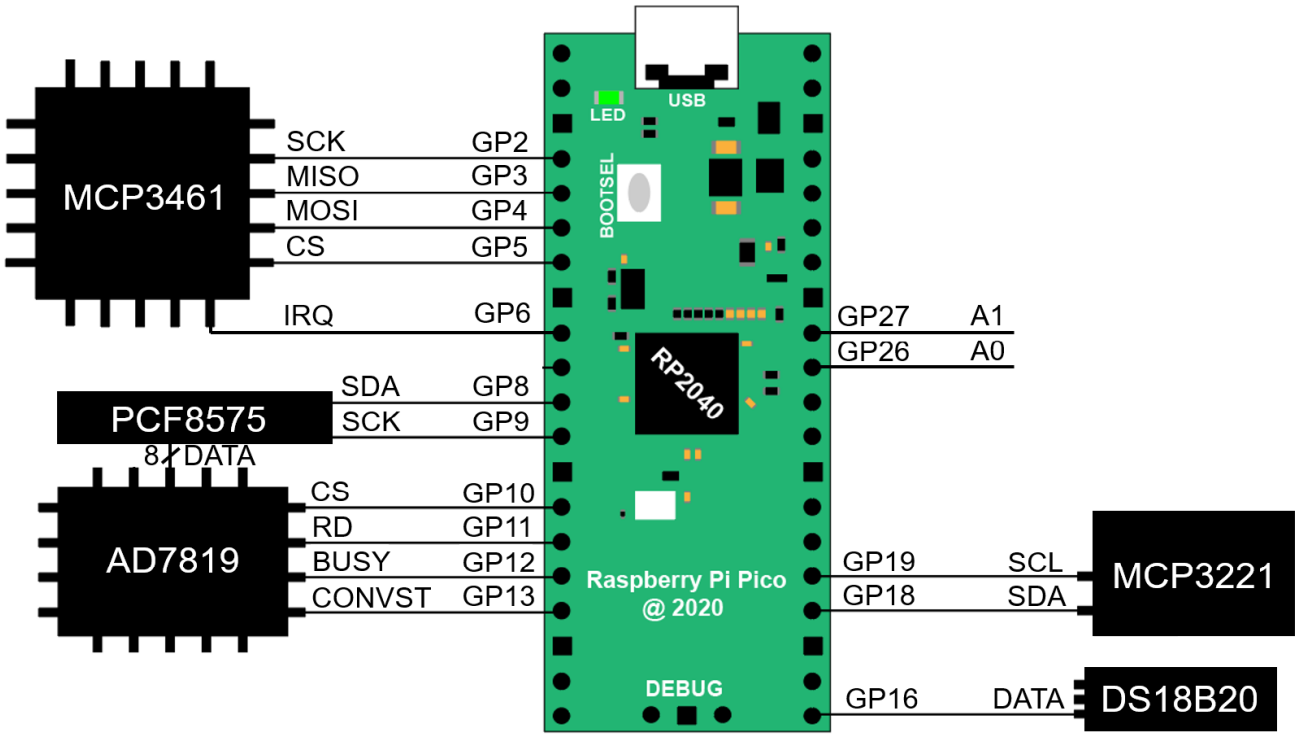
Rys. 22. Schemat bloku jednostki centralnej



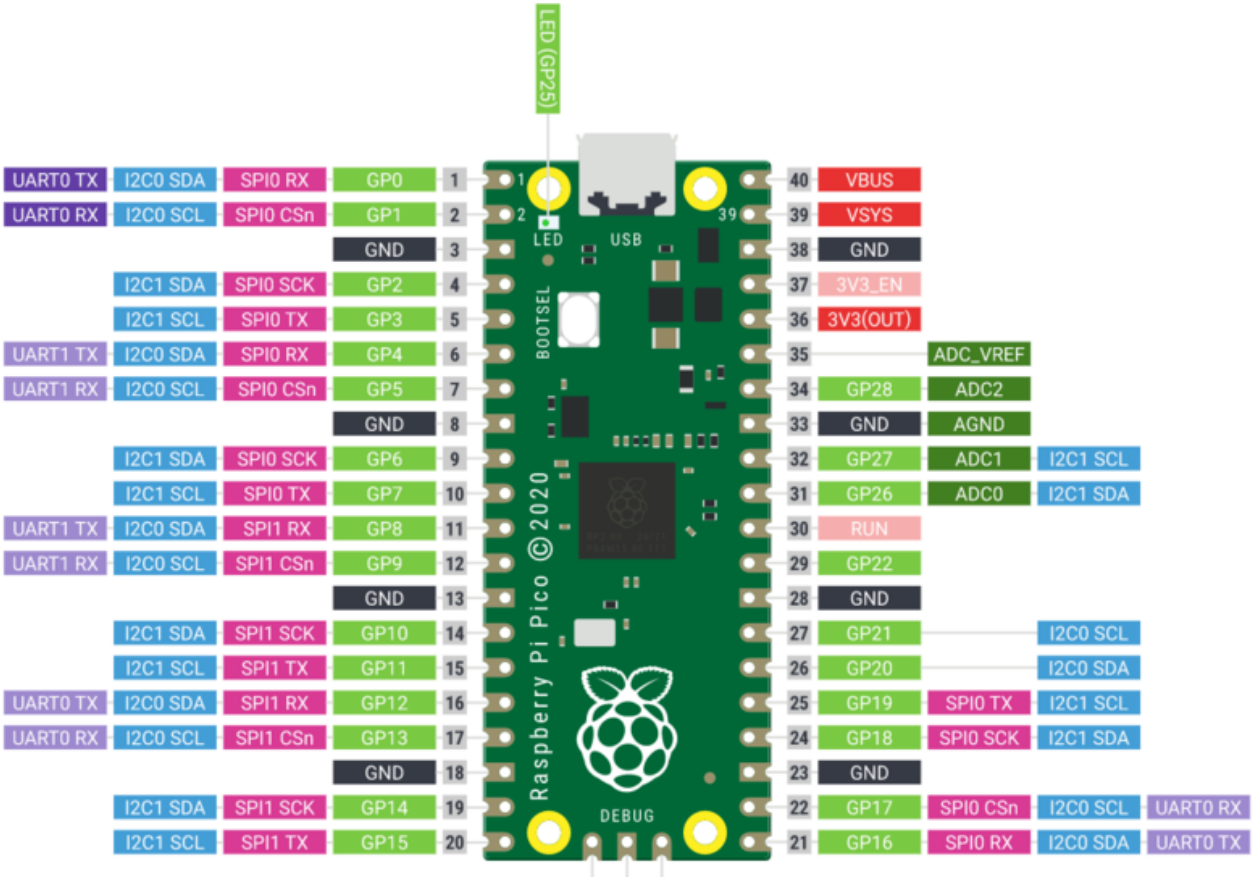
Rys. 23. Schematy bloku przetworników.



Rys. 24. Widok ogólny makiety z zaznaczoną lokalizacją kluczowych elementów



Rys. 25. Uproszczony diagram wskazujący na kluczowe połączenia układów ADC z Raspberry Pico



Rys. 26. Opis portów komunikacyjnych platformy Raspberry Pico

5. REALIZACJA ĆWICZENIA

W ćwiczeniu poznać można różne sposoby przygotowania sygnału analogowego doprowadzanego do wejść analogowych przetworników A/C oraz zaznajomić się z różnymi metodami sterowania i komunikacji cyfrowej z przetwornikami A/C

5.1. PRZYGOTOWANIE WEJŚĆ ANALOGOWYCH

Tor wejść analogowych może korzystać z wewnętrznego źródła napięcia ustawianego potencjometrem albo przyjmować napięcia z zewnątrz przez złącze IN1 i odpowiednio je wzmocnić, osłabić czy dodać składową stałą.

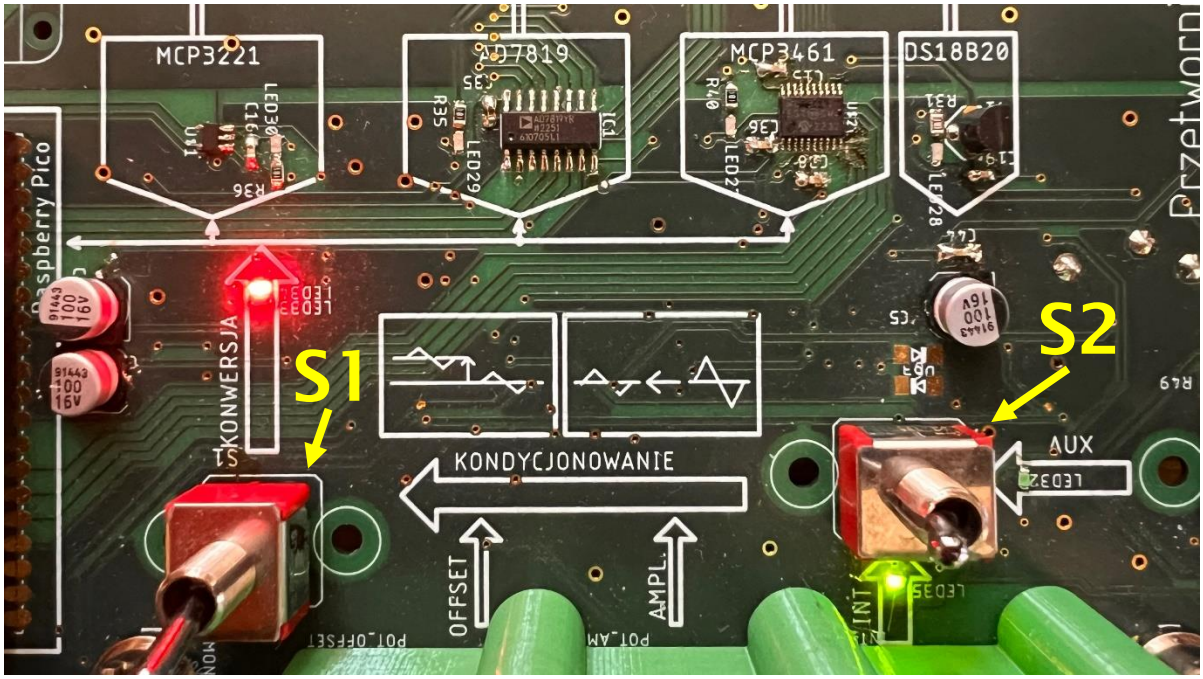
5.1.1. Wariant z wewnętrznym źródłem napięcia

1. Przed rozpoczęciem ćwiczenia upewnić się że:

   - potencjometry ustawione są na najniższym poziomie (ustawienie przeciwne do ruchu wskazówek zegara),

   - przełącznik S2 powinien znajdować się w położeniu aktywującym sygnał wewnętrzny (dźwignia przełącznika ustawiona w lewo), a przełącznik S1 powinien znajdować się w położeniu dezaktywującym doprowadzenie sygnału (dźwignia przełącznika ustawiona w kierunku zespołu potencjometrów). Widok prawidłowej konfiguracji przedstawiono na poniższym zdjęciu. Po zasileniu płytki ustawienia przełączników powinny być sygnalizowane odpowiednio: dla przełącznika S2 - świeci zielona dioda od strony potencjometrów „INT”, dla przełącznika S1 - świeci czerwona dioda na strzałce pokazującej kierunek przetwarzania sygnałów „KONWERSJA”.

   UWAGA – dźwignie przełączników w położeniu środkowym nie załączają żadnego sygnału.



Rys. 27. Widok wstępnej konfiguracji makiety.

2. Do wyjścia monitorującego należy podłączyć woltomierz.
3. Zasilić makietę napięciem symetrycznym +/- 12V. Obecność napięcia zasilającego sygnalizowana jest świeceniem trzech czerwonych diod w bloku zasilania makiety.
4. Potencjometrem WEW (po prawej stronie) ustalić wartość zadaną przez prowadzącego (sugerowana wartość: ok. 1V), obserwując wskazania na przyrządzie.
5. Po stwierdzeniu przez prowadzącego uzyskania właściwych ustawień, przełącznik S1 ustawić w położeniu załączenia. Spowoduje to doprowadzenie sygnału do wejść przetwornika. Sygnalizowane to będzie załączeniem zielonej diody w obszarze strzałki „KONWERSJA".

5.1.2. Wariant z zewnętrznym źródłem napięcia

1. Przed rozpoczęciem ćwiczenia upewnić się że:

   - potencjometry ustawione są na najniższym poziomie (ustawienie przeciwne do ruchu wskazówek zegara),

   - przełącznik S2 powinien znajdować się w położeniu aktywującym sygnał zewnętrzny (dźwignia przełącznika ustawiona w prawo), a przełącznik S1 powinien znajdować się w położeniu dezaktywującym doprowadzenie sygnału (dźwignia przełącznika ustawiona w kierunku zespołu potencjometrów). Przełącznik S3 powinien znajdować się w położeniu

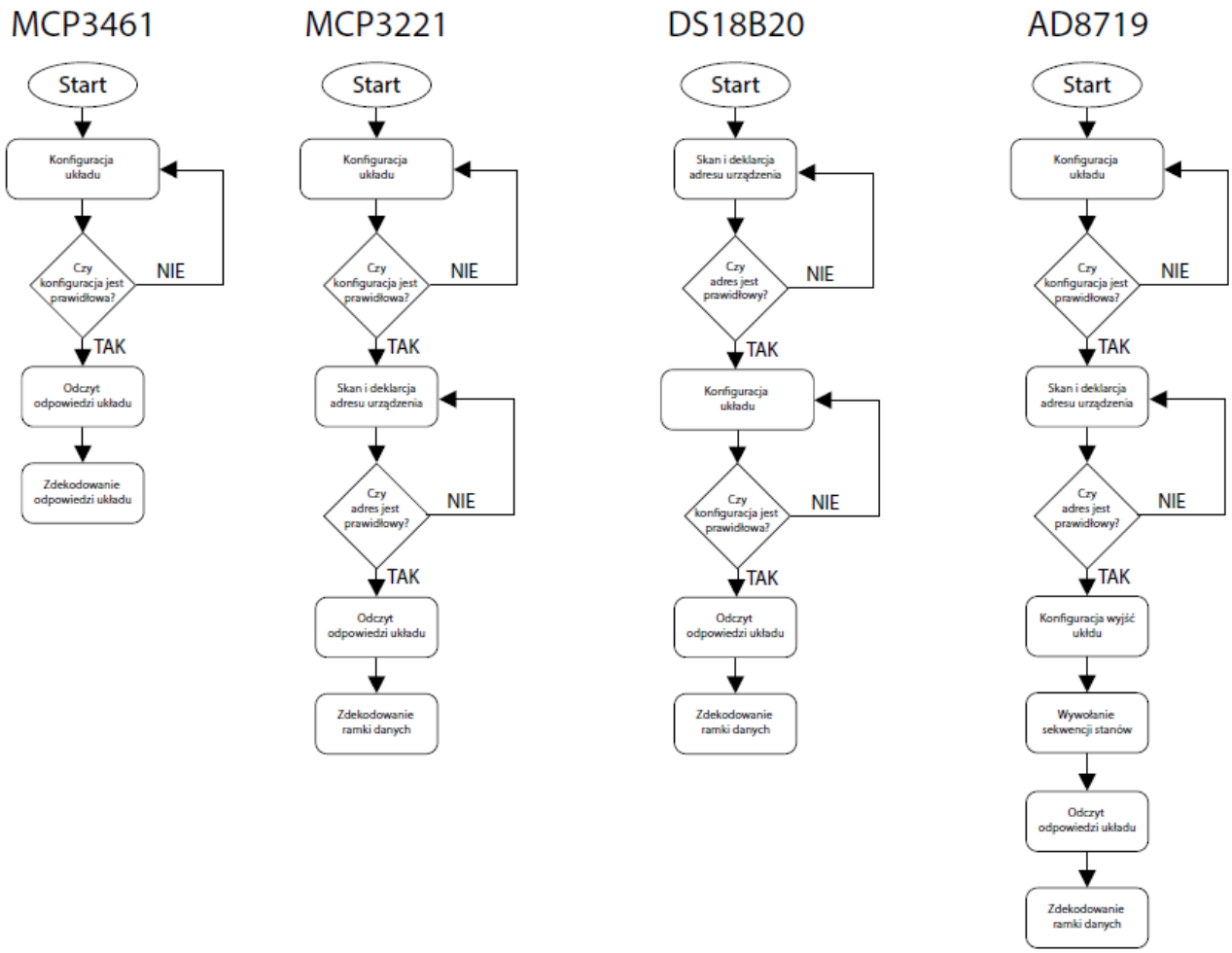
aktywującym sygnał z wejścia INT1 (dźwignia przełącznika ustawiona w dół). Ustawienia przełączników powinny być sygnalizowane odpowiednio: dla przełącznika S3 - świeci zielona dioda od strony wejścia „IN1", dla przełącznika S2 - świeci zielona dioda od strony wejść „AUX", dla przełącznika S1 - świeci czerwona dioda na strzałce pokazującej kierunek przetwarzania sygnałów „KONWERSJA".

2. Ustawić podane przez prowadzącego parametry sygnału (sugerowany sygnał symetryczny, pk-pk 2 V, sinus, 10 Hz).
3. Podłączyć oscyloskop do wyjścia monitorującego. UWAGA! Wejście oscyloskopu musi pracować w trybie DC.
4. Potencjometrem *Amplitude* ustawić tak amplitudę sygnału obserwowanego na wyjściu, aby mieściła się on w przedziale napięć przetwarzanych przez przetworniki (wartość pk-pk nie może przekraczać 3V).
5. Potencjometrem *Offset* ustawić poziom napięcia przesunięcia, aby sygnał obserwowany znalazł się w oknie napięciowym przetwarzania (sygnał musi mieścić się w zakresie od 0V do 3V).
6. Po stwierdzeniu przez prowadzącego uzyskania właściwych ustawień, przełącznik S3 ustawić w położeniu do góry. Spowoduje to doprowadzenie sygnału do wejść przetwornika i pozwoli na przeprowadzenie przetwarzania sygnału.

5.2 WYKONYWANE POMIARY

1) Podłączyć moduły Raspberry Pico do komputera za pomocą przewodu USB. Na komputerze, korzystając ze środowiska Thonny należy utworzyć nowe pliki skryptów związanych z wykonywaniem poszczególnych części ćwiczenia. Polecenia można też wydawać do wykonania w trybie natychmiastowym w konsoli programu Thonny. Upewnić się należy przedtem, że w prawej dolnej części programu Thonny będzie sygnalizowane uruchamianie kodu na Raspberry Pico (powinien wyświetlać się komunikat: MicroPython (Raspbery Pi Pico)). Dla poszczególnych przetworników wykonać pomiary DC zgodnie z poniżej opisanymi protokołami. Dla wskazanego przetwornika wykonać pomiar AC.

2) Dla napięcia DC generowanego przez układ na płytce (wewnętrzne źródło napięcia), uzyskać konwersję układów wskazanych przez prowadzącego. Ustawić i dokonać przetworzenia 10 wartości napięcia w całym zakresie przetwarzania, notując wartość napięcia oraz wynik przetwarzania. Sporządzić charakterystyki przetwarzania (wartości kodu w funkcji napięcia). Zakładając, że zakres przetwarzania zawiera się w przedziale 0 – 3 V, przeliczyć zarejestrowane wartości na napięcie i zweryfikować poprawność wskazań. Ocenić liniowość przetwarzania. Wyznaczyć rozdzielczość przetwarzania (w woltach).
3) Ustalić na generatorze sygnałowym przebieg sinusoidalny 10 Hz, 2 V pk-pk, symetryczny. Przy użyciu elementów kondycjonujących, ustalić właściwy zakres napięć wejściowych na przetwornika. Wprowadzając pętle dokonywać próbkowania z zadaną latencją w pętli (0,1; 1; 5; 10 i 20 sekund). Dla każdego wariantu liczbę próbek dobrać tak, aby zarejestrować nie mniej niż trzy okresy przebiegu. Porównać wyniki przetwarzania dla zastosowanych latencji oraz dla poszczególnych układów zestawiając je ze sobą na wykresach (przebiegi czasowe).
4) W przypadku termometrów, po uzyskaniu wyników przetwarzania, należy dokonać zdekodowania (przeliczenia) i ocenić wiarygodność wyniku. Po stwierdzeniu prawidłowości odczytów, można powtórzyć przetwarzanie po wcześniejszym ogrzaniu czujnika palcem.
5) Sposoby komunikacji w uproszczony sposób przedstawione są na rysunku 27, natomiast przed przystąpieniem do ćwiczenia konieczne jest zapoznanie się ze specyfiką działania poszczególnych interfejsów oraz ich obsługi i konfiguracji w przypadku obsługiwanych układów. W konfiguracji interfejsów można wykorzystać rysunki 25 i 26, ilustrujące kluczowe połączenia między układami.



Rys. 27. Uproszczone diagramy komunikacji z poszczególnymi układami.

Należy zwrócić uwagę, że środowisko zostało przygotowane w taki sposób, aby w poleceniu obsługującym dany układ, znalazła się jego nazwa. Ułatwia to przeprowadzanie poszczególnych działań konfiguracyjnych i obsługę procesu przetwarzania sygnałów.

Do prawidłowej konfiguracji układów, wymaganej w ćwiczeniu konieczna jest znajomość kluczy i identyfikatora fizycznych wyprowadzeń platformy Pi Pico. Klucz może przyjąć jedną z 2 wartości (0 i 1). Numer klucza został opisany odpowiednim numerem bezpośrednio po nazwie protokołu komunikacyjnego.

Rys. 28. Lokalizacja informacji o kluczu protokołu komunikacyjnego

5.3 KOMUNIKACJA Z PRZETWORNIKAMI

5.3.1. PRZETWORNIK ANALOGOWO – CYFROWY Z MAGISTRALĄ I2C – MCP3221

Na podstawie rysunków 25 i 26 określ, do których wyprowadzeń Raspberry Pi Pico dołączone są linie magistrali I2C przetwornika MCP3221 (numery x wejść GPx), oraz który z interfejsów I2C odpowiedzialny będzie za komunikację (I2Cx). Zapisz to w sprawozdaniu.

Utwórz nowy skrypt w środowisku Thonny, pozwalający na komunikację z wykorzystaniem magistrali I2C:

```
import machine
#importowanie bibliotek obsługujących wejścia (Pin) i magistralę I2C
from machine import Pin, I2C
#utworzenie obiektu i2c klasy I2C obsługującego interfejs
i2c = I2C(i2c_num, scl=Pin(pin_scl), sda=Pin(pin_sda), freq=100000)
#odczyt adresow urzadzen na magistrali I2C
addresses = i2c.scan()
print (addresses)
```

W miejsce **i2c_num**, **pin_scl**, **pin_sda** wpisz odpowiednio: numer interfejsu I2C, numer wejścia GPx linii SCL oraz numer wejścia linii SDA, do których dołączony jest przetwornik MCP3221.

Metoda **scan()** w klasie I2C zwraca tabelę zawierającą adresy urządzeń dołączonych do magistrali. Jeżeli dobrze jest wszystko skonfigurowane, polecenie **print** powinno spowodować wyświetlenie tabeli zawierającej jeden element o wartości takiej, jak adres przetwornika. Odnajdź na makiecie układ MCP3221 i posługując się tabelą 7 sprawdź, czy zwrócony adres przetwornika jest właściwy. Wnioski zapisz w sprawozdaniu.

Na rysunku 14 pokazany jest sposób transmisji danych w tym przetworniku. Po wysłaniu do niego jego adresu przetwornik odpowiada dwoma bajtami zawierającymi wynik przetwarzania. Jeśli przetwornik na swoje wejście otrzymuje odpowiednie napięcie, to można je przetworzyć na postać cyfrową uzupełniając dotąd powstały skrypt o polecenie:

```
#odczyt 2 bajtów
sample = i2c.readfrom(addresses[0],2);
```

Polecenie to powoduje odczyt 2 bajtów (2. argument metody **i2c.readfrom**) z układu o adresie takim, jak pierwszy element tabeli **addresses**, która była odczytana uprzednio. Zmienna **sample** zawierać będzie 2 bajty z wynikiem przetwarzania.

Zastanów się jak to zrobić a następnie uzupełnij skrypt o:

- obliczenie wartości słowa z wynikiem przetwarzania na podstawie dwóch bajtów zawartych w elementach tabeli **`sample`**: **`sample[0]`** i **`sample[1]`**
- przeliczenie wartości słowa z wynikiem przetwarzania na napięcie; weź pod uwagę wartość napięcia odniesienia równą 3,3 V.

Jeśli potrzebne będzie odczytywanie większej liczby próbek, niż jedna, jest możliwe również wymuszenie pomiarów w trybie ciągłym. Zwróć uwagę na rysunek 15. Wymaga to po prostu odczytywania większej niż 2xN bajtów z przetwornika aby odczytać N próbek. O szybkości próbkowania będzie decydowała częstotliwość pracy magistrali I2C, którą można zmieniać podając inną wartość **`freq`** w konstruktorze obiektu klasy **`I2C`**.

5.3.2. CZUJNIK TEMPERATURY DS18B20 Z INTERFEJSEM ONEWIRE

Innym przetwornikiem dostępnym na makiecie jest popularny cyfrowy czujnik temperatury DS18B20. W nim komunikacja odbywa się za pomocą magistrali OneWire. Na podstawie rysunków 25 i 26 określ, do których wyprowadzeń Raspberry Pi Pico dołączony jest przetwornik DS18B20. Zapisz to w sprawozdaniu.

Nawiązanie komunikacji za pomocą interfejsu OneWire w środowisku MicroPython wymaga użycia poleceń wymienionych poniżej:

```
#importowanie niezbednych bibliotek
import machine, onewire, ds18x20, time
from machine import Pin
from onewire import OneWire
from ds18x20 import DS18X20
#utworzenie obiektu obslugujacego magistrale onewire
ow = OneWire(Pin(pin_ow))
#uwtorzenie obiektu klasy narzędziowej DS18X20 zawierajacej metody do jego obslugi
ds = DS18X20(ow)
#odczyt adresow urzadzen dolaczonych do magistrali onewire
addresses = ow.scan()
print(addresses)
```

Podobnie, jak poprzednio, program ten po zaimportowaniu odpowiednich bibliotek potrzebuje podania numeru wejścia GPx, którym obsługiwana jest magistrala pin_ow.

Po przeskanowaniu magistrali metodą **`ow.scan()`** otrzymuje się tabelę zawierającą adresy urządzeń podpiętych do magistrali OneWire. W wypadku makiety, powinna ona zawierać jeden element, którego wartość polecenie **`print`** wyświetli w konsoli.

Dalej, polecenie

```
ds.convert_temp()
```

spowoduje spróbkowanie bieżącej wartości wskazywanej przez wewnętrzny czujnik temperatury we wszystkich układach DS18B20 podłączonych do magistrali (gdyby było ich więcej). Trwa to chwilę stąd pisząc automatyczny skrypt poleceniem

```
time.sleep_ms(750)
```

powodujemy opóźnienie jego wykonania o 750 ms. Po tym można odczytać wynik przetwarzania poleceniami:

```
data = ds.read_scratch(addresses[0])
temp = ds.read_temp(addresses[0])
```

po czym w data znajdzie się „scratch pad" czujnika zawierający szereg bajtów, zgodnie z rysunkiem 4, a w temp wartość temperatury w stopniach Celsjusza. Spróbuj dodając do skryptu odpowiednie polecenia **`print`** wyświetlić w konsoli wartość cyfrową słowa odczytanego z czujnika i temperaturę. Zapisz te wartości w sprawozdaniu. W dokumentacji czujnika znajdź zależność pomiędzy jedną a drugą wielkością i uzasadnij prawidłowość wyliczenia temperatury przez funkcje biblioteki obsługującej DS18B20.

Odczyt można ponowić kilkukrotnie ogrzewając czujnik palcem i zmieniając tym samym jego wskazania.

5.3.3. PRZETWORNIK Z ODCZYTEM RÓWNOLEGŁYM AD7819

Przetwornik AD7819 dołączony jest do Raspberry Pi Pico częściowo bezpośrednio a częściowo z wykorzystaniem tzw. ekspandera I2C/8 bit, zgodnie z rysunkami 25 i 26. Zanotuj, do których wyprowadzeń Raspberry Pi dołączone są linie CS, RD CONVST sterujące przetwornikiem, linia BUSY wskazująca zajętość przetwornika, oraz do których linii dołączone są SDA i SCL ekspandera I2C/8 bit i do którym interfejsem I2C są obsługiwane. Zapisz te wartości w sprawozdaniu.

Przygotowanie połączenia z przetwornikiem wymaga wykonania skryptu, którego początek prezentuje się następująco:

```
import machine
from machine import I2C, Pin

# obiekty klasy Pin do obslugi linii sterujacych
cs = Pin(pin_cs, Pin.OUT, value = ...)
rd = Pin(pin_rd, Pin.OUT, value = ...)
convst = Pin(pin_convst, Pin.OUT, value = ...)
```

w którym definiujesz trzy obiekty klasy Pin za pomocą których będzie można zmieniać stan logiczny na wyprowadzeniach Raspberry Pi Pico, które pełnić będą rolę wyjść cyfrowych (określa się to określając rolę wyprowadzenia **`Pin.OUT`**). Podać należy zamiast **`pin_cs`**, **`pin_rd`**, **`pin_convst`** numery portów GPx a używając opcjonalnego parametru **`value`** i zamiast wielokropka wpisując 0 albo 1 można określić, jaki początkowy stan logiczny ma mieć to wyprowadzenie. Spójrz na rysunek 17 i na jego podstawie określ, jakie to mają być wartości dla tych trzech sygnałów. Dalej:

```
busy = Pin(pin_busy, Pin.IN)
```

utworzy obiekt **`busy`** klasy Pin, który wyprowadzenie o odpowiednim numerze GPx skonfiguruje jako wejście cyfrowe za pomocą którego można stwierdzać, czy przetwornik jest zajęty. Z ekspanderem I2C połączyć się można magistralą I2C w sposób już Ci znany:

```
i2c = I2C(i2c_num, scl=Pin(pin_scl), sda=Pin(pin_sda), freq=400000)
addresses = i2c.scan();
```

gdzie oczywiście należy podać odpowiednie numery interfejsu I2C oraz wyprowadzeń GPx. Ekspander pracuje z maksymalną częstotliwością magistrali 400 kHz. Przeskanowanie magistrali powinno zwrócić w tabeli **`addresses`** jedną wartość odpowiadającą adresowi ekspandera.

W dalszej części ćwiczenia na podstawie rysunku 17 przetwórz wartość napięcia podanego na wejście przetwornika w sposób opisany w punkcie 5.1. Zapisz w sprawozdaniu wartość lub wartości napięć, jakie były stosowane.

Do sterowania i komunikacji z przetwornikiem masz do dyspozycji metody: **`on()`** i **`off()`** w obiektach klasy Pin, np. **`cs.off()`** czy **`convst.on()`**, które zmieniają stany sterujących przetwornikiem: on na wysoki, off na niski. Odczyt stanu logicznego linii busy można uzyskać za

pomocą metody **busy.value()**, która zwraca 0 lub 1 zależnie od stanu logicznego. Natomiast odczytanie danych z interfejsu równoległego przetwornika za pośrednictwem ekspandera odbywa się następująco:

```
data = i2c.readfrom(addresses[0], 2)
```

o ile **addresses** to tabela adresów urządzeń na magistrali I2C uzyskana wcześniej. Odczyt danych z ekspandera wymaga odczytu dwóch bajtów, stąd wartość 2 jako drugi argument. W **data** znajdą się dwa odczytane bajty, z których **data[0]** zawiera wynik przetwarzania.

Używając tych metod odtwórz sekwencję zmian stanów logicznych z rysunku 17. Możesz próbować najpierw wywoływać je „ręcznie" wpisując je w konsolę i obserwując świecenie diod LED wskazujących stan linii magistrali. Ostatecznie stwórz skrypt pozwalających zainicjować i odczytać wynik przetwarzania.

Odczytaj bajt z wynikiem przetwarzania dla kilku napięć. Biorąc pod uwagę napięcie odniesienia równe 3,0 V oblicz wynik przetwarzania w woltach. Porównaj go ze wskazaniami woltomierza.

5.3.4. PRZETWORNIK WEWNĘTRZNY RASPBERRY PICO

Aktywacja wewnętrznego przetwornika Raspberry PICO odbywa się następująco:

```
analog_value = machine.ADC(port wejściowy)
```

gdzie parametr **port wejściowy** należy odczytać ze schematu połączeń, sprawdzając do których wejść doprowadzony jest sygnał.

Przeprowadzenie przetwarzania wywoływane jest poleceniem:

```
reading = analog_value.read_u16()
```

5.3.5. TERMOMETR WEWNĘTRZNY RASPBERRY PICO

Aktywacja wewnętrznego przetwornika Raspberry PICO odbywa się następująco:

```
from machine import ADC
temp_sensor = ADC(4)
```

Natomiast przeprowadzenie przetwarzania wywoływane jest poleceniem:

```
temp_int = temp_sensor.read_u16()
```

Uzyskana w ten sposób wartość jest jedynie wynikiem przetwarzania spadku napięcia na złączu P-N. Dlatego też, aby otrzymać wartość temperatury w skali Celsjusza, konieczne jest przeprowadzenie obliczeń:

```
spadek_napiecia = temp_int * (3.3 / 65535.0)

temperatura_celsjusz = 27 - (spadek_napiecia - 0.706) / 0.001721
```

Dysponując powyższymi informacjami oraz wynikami pomiarów, wyznaczyć rozdzielczość pomiaru temperatury i odnieść te informacje do parametrów przetwornika DS18B20.

Wykonać pomiary oboma przetwornikami w możliwie krótkim odstępie czasowym. Od czego mogą zależeć różnice w uzyskanych wynikach?

PROTOKÓŁ

CZŁONKOWIE GRUPY	
1.	
2.	
3.	
4.	
5.	
6.	

NR GRUPY	NUMER ĆWICZENIA	DATA ZAJĘĆ	PODPIS PROWADZĄCEGO