International Scientific Conference 20 UNITECH 26

INTERNATIONAL SCIENTIFIC CONFERENCE
19-21 November 2026
GABROVO

PERIPHERALS FOR MEASUREMENT AND CONTROL: ARCHITECTURES AND AN RP2350 DATA ACQUISITION CASE STUDY

Krasimir Gichev

Technical University of Gabrovo, Department of Electronics, 4 Hadji Dimitar Str., 5300 Gabrovo, Bulgaria
E-mail: k.gichev@tugab.bg

Abstract
Measurement and control systems often need digital interfaces that fixed microcontroller peripherals do not support well, such as framed converter interfaces with data-ready signalling. Implementation options range from software bit-banging through programmable I/O subsystems to FPGAs and ASICs, differing in timing determinism, flexibility, unit cost and non-recurring engineering effort. This paper classifies these options and compares programmable I/O subsystems, emphasising the RP2040/RP2350 PIO and the TI PRU. In the case study, an RP2350 PIO state machine with chained DMA reads a four-channel, 24-bit, 128 kSPS delta-sigma ADC (TI ADS131A04) with one CPU interrupt per 1024 frames. An analytical timing model relates the ADC clock dividers to the data rate. It also relates the PIO divider and program to the serial-clock period, frame duration, sampling instants and setup/hold margins, including the wiring. Oscilloscope measurements of three firmware versions agree with the model to within 2 ns. It shows that the original clock ran at the ADC's minimum specified period and reveals a one-bit data misalignment invisible in the waveforms. Both findings were confirmed and corrected on hardware: a configuration with 33 % clock margin read 76.8 million frames without a readout fault. The system serves drive energy measurements.

Keywords: programmable I/O, RP2350 PIO, PRU-ICSS, timing analysis, data acquisition.

INTRODUCTION

Measurement and control equipment for electric drives exchanges data with multi-channel ADCs, position encoders and industrial networks through serial interfaces that are only partly standard. A simultaneous-sampling delta-sigma ADC signals each conversion with a data-ready (/DRDY) line and expects a complete multi-word frame to be read before the next result [1]; encoder protocols such as EnDat 2.2 add strict bit timing and protocol-specific framing [2]. The fixed-function SPI, I²C and UART controllers of a microcontroller (MCU) meet such requirements only with interrupt or polling code, whose latency adds to the interface timing.

Designers traditionally choose between software emulation on GPIO pins ("bit-banging"), which costs CPU time and has interrupt-dependent jitter [3], and an FPGA or ASIC, which adds a device and a toolchain or, for an ASIC, very high non-recurring engineering (NRE) cost [4]. *Programmable I/O subsystems* occupy the middle ground: the Raspberry Pi PIO [5], the TI Programmable Real-time Unit (PRU) [6] and NXP FlexIO [7] are small deterministic engines next to the pins and the DMA system. Because every instruction takes a known number of cycles, the interface timing can be computed from the program and clock configuration before any measurement.

This paper compares the implementation options and programmable I/O subsystems from primary documentation, and builds and checks an analytical timing model of an

This is an open access article licensed under Creative Commons Attribution 4.0 International
Print ISSN: 1313-230X, Online ISSN: 3033-1404

RP2350 PIO interface to a 24-bit, 128 kSPS, four-channel ADC, from the clock registers to bit-level setup and hold margins. The system belongs to the author's doctoral research on the efficiency of electric drive systems, which needs simultaneous voltage and current measurement for energy profiles of production operations [8].

PROGRAMMABLE I/O: OPTIONS AND SUBSYSTEMS

Timing-critical I/O can be implemented at six levels. *Bit-banging* needs no extra silicon, but its timing is set by interrupts and wait states [3]. *Fixed-function peripherals* support DMA only for anticipated protocols; framing tied to an event such as /DRDY needs CPU action per frame. *Configurable logic cells* such as the Microchip CLB [9] suit glue logic but lack sequencing and data paths. *Programmable I/O subsystems* have cycle-predictable engines, shift registers, FIFOs and DMA interfaces, a timing resolution of one engine cycle (typically 3–10 ns) and essentially no marginal cost, as they are already inside inexpensive MCUs. An *FPGA* adds a second device and a synthesis toolchain. A *custom ASIC* has the lowest unit cost at volume, but its masks alone cost about US$0.5M at 55/65 nm and the total NRE is several times that [4].

The subsystems fall into three classes (Table 1): (a) logic-configurable blocks [9]; (b) narrow I/O state machines with shift registers and FIFOs, the PIO [5], [10] and FlexIO [7]; (c) I/O co-processor cores with own memories, the TI PRU [6] and the NXP eTPU [11].

Table 1. *Representative programmable I/O subsystems (vendor-documented parameters)*

Subsystem (vendor)	Class	Engines	Program memory	Max clock
PIO, RP2040 / RP2350 (Raspberry Pi) [10], [5]	b	2 / 3 blocks × 4 SM	32 instr. per block	133 / 150 MHz
PRU-ICSS, AM335x (TI) [6]	c	2 PRU	8 KB IRAM per core	200 MHz
PRU_ICSSG, AM64x/AM243x (TI) [6]	c	2 × (2 PRU + 2 RTU_PRU + 2 TX_PRU)	12 / 8 / 6 KB	333 MHz
FlexIO, Kinetis, i.MX RT (NXP) [7]	b	e.g. 8 shifters, 8 timers	configuration only	—
CLB, PIC16F13145 (Microchip) [9]	a	32 LUT elements	bitstream	—
eTPU, MPC5554 (NXP) [11]	c	2 engines × 32 channels	up to 16K micro-instr.	—

RP2040/RP2350 PIO. The RP2350 has three PIO blocks of four state machines, each block with a shared 32-instruction memory [5]. Nine instructions (JMP, WAIT, IN, OUT, PUSH, PULL, MOV, IRQ, SET) execute in one cycle each, with side-set pins and delay cycles. Each state machine has shift registers, 4-word FIFOs, a fractional clock divider and a DMA interface. Every GPIO input passes a two-flip-flop synchroniser (two cycles of latency) that can be bypassed per pin [5]; this matters in the model below.

TI PRU. The PRU is a pipeline-free 32-bit RISC core with deterministic execution and single-instruction access to dedicated pins [6]. The PRU_ICSSG of the AM64x/AM243x adds hardware input modes for EnDat 2.2 and nine-channel sigma-delta demodulation [6], i.e. the encoder and current-sensing interfaces of servo drives, but exists only in application processors that cost considerably more than a small MCU.

Comparison. Both are deterministic: a PIO state machine is a configurable shifter that relies on DMA, whereas a PRU can run complete protocol state machines and buffering.

CASE STUDY: PIO-BASED ACQUISITION FROM A 24-BIT DELTA-SIGMA ADC

The front end is the TI ADS131A04 evaluation module: four simultaneous-sampling 24-bit channels at up to 128 kSPS [1]. A Raspberry Pi Pico 2 (RP2350, 150 MHz, Pico SDK 2.2.0) connects to it through six signals on GPIO 16–21 over unshielded jumper wires about 200 mm long (Fig. 1). The hardware SPI0 controller first configures the ADC; the pins are then

handed to PIO0 state machine 0, which runs the capture program (Fig. 2, original 12 instructions without the marked lines): it waits for /DRDY high and then low, asserts /CS, clocks 5 × 24 bits (status plus four channels) on SCLK via side-set while sampling DOUT, pushes each word to the RX FIFO and releases /CS. Two chained DMA channels alternately fill two 1024-frame buffers, so the CPU handles one interrupt every 8 ms instead of 128,000 per second.

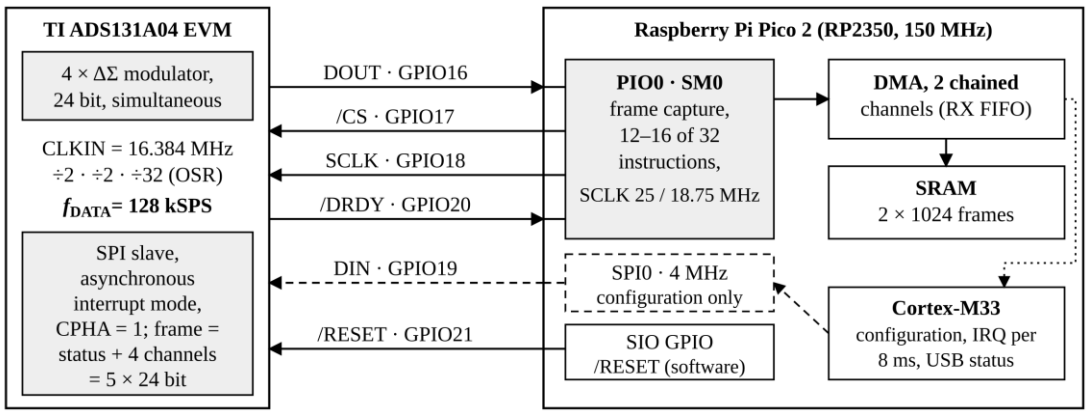


Fig. 1. *Block diagram of the acquisition system.*

```
  .side_set 1  ; SCLK
  .wrap_target
    wait 1 gpio 20 side 0 ; DRDY high
    wait 0 gpio 20 side 0 ; DRDY low
    set pins, 0    side 0 ; CS low
    set y, 4       side 0 ; 5 words
  word_loop:
~   set x, 22      side 0 ; was 23
+   nop            side 1 [1]
+   nop            side 0 [1]
  bit_loop:
    in pins, 1     side 1 ; sample
    nop            side 1
    nop            side 0
    jmp x-- bit_loop side 0
+   in pins, 1     side 0 ; LSB
    push block     side 0 ; to DMA
    jmp y-- word_loop side 0
    set pins, 1    side 0 ; CS high
  .wrap
```

Fig. 2. *PIO capture program after the bit-alignment fix (15 instructions): + added, ~ changed (23 → 22) relative to the original 12 instructions; the 18.75 MHz variant is in [12].*

ANALYTICAL TIMING MODEL

Step 1 — ADC clock chain and data rate

Following the datasheet [1], the ADC divides its input clock by $N_{CLK} = 2$ and $N_{ICLK} = 2$ and the modulator output by OSR = 32 (CLK1/CLK2 registers). With f_{CLKIN} = 16.384 MHz

$$f_{DATA} = \frac{f_{CLKIN}}{N_{CLK}\, N_{ICLK}\, OSR} = \frac{16.384\ \text{MHz}}{2 \cdot 2 \cdot 32},$$

$$f_{DATA} = 128\ \text{kSPS}, \quad T_{DATA} = 7.8125\ \mu\text{s} \tag{1}$$

Each frame has $N_w = 5$ words of $W = 24$ bits (status plus four channels; default mode, CRC off), i.e. 120 bits.

Step 2 — PIO clock and SCLK

The state machine runs at $f_{PIO} = f_{sys}/D$ with f_{sys} = 150 MHz (T_{sys} = 6.667 ns) and D = 1.5, i.e. 100 MHz on average. A fractional divider alternates between $\lfloor D \rfloor$ and $\lceil D \rceil$ system cycles [5], so n PIO cycles are jitter-free if nD is an integer. The program spends $k = 4$ PIO cycles per bit, $h = 2$ of them with SCLK high:

$$T_{SCLK} = kD\,T_{sys} = 40.0\ \text{ns}\ (f_{SCLK} = 25\ \text{MHz}),$$

$$T_H = T_L = hD\,T_{sys} = 20.0\ \text{ns} \tag{2}$$

Because $hD = 3$ is an integer, both phases are exactly 20 ns. The datasheet requires $t_{c(SC)} \geq 40$ ns and $t_{w(SCHL)} \geq 20$ ns [1], so the original interface meets both limits with **zero margin**.

Step 3 — Frame timing and feasibility

Each word takes $k \cdot W + o_w$ PIO cycles, with $o_w = 3$ (SET X, PUSH, JMP Y); $n_0 = 2$

cycles (SET PINS, SET Y) precede the first word. The /CS-low interval spans

$$N_{\mathrm{CS}} = n_0 + N_{\mathrm{w}}(kW + o_{\mathrm{w}}) = 497,$$
$$T_{\mathrm{CS}} = N_{\mathrm{CS}}\, D\, T_{\mathrm{sys}} = 4.97\ \mu\mathrm{s} \qquad (3)$$

The bit-alignment fix adds one cycle per word ($o_{\mathrm{w}} = 4$, $N_{\mathrm{CS}} = 502$, 5.02 µs); the $k = 8$ program of the final firmware has $o_{\mathrm{w}} = 3$ and $n_0 = 3$ ($N_{\mathrm{CS}} = 978$, 6.52 µs).

The latency from /DRDY↓ to /CS↓ consists of the two-cycle input synchroniser, its sampling phase (0–1 cycle), the divider enable (0–1 cycle) and one SET instruction:

$$3T_{\mathrm{sys}} \le t_{\mathrm{DC}} - t_{\mathrm{pi}} - t_{\mathrm{po}} \le 5T_{\mathrm{sys}} \qquad (4)$$

where t_{pi} and t_{po} are the input and output pad delays. With $D = 1$ the divider-enable term disappears and the upper bound falls to 4 T_{sys}, so the spread of t_{DC} is 2 T_{sys} for $D = 1.5$ and 1 T_{sys} for $D = 1$. After /CS↑ the ADC returns /DRDY high after $t_{\mathrm{p(CSDR)}} = 2/f_{\mathrm{ICLK}} = 244$ ns [1]. The frame is feasible if

$$t_{\mathrm{DC}} + T_{\mathrm{CS}} + t_{\mathrm{p(CSDR)}} + t_{\mathrm{sync}} \le T_{\mathrm{DATA}} \qquad (5)$$

with $t_{\mathrm{sync}} = 3\ T_{\mathrm{sys}} = 20$ ns to detect /DRDY high. With t_{DC} at its upper bound, 5 $T_{\mathrm{sys}} + p = 39.8$ ns, the left-hand side of (5) is 5.27 µs, leaving 2.54 µs (32 %) of slack.

Step 4 — Bit sampling with cable delays

The ADC (CPOL = 0, CPHA = 1) drives bit j after SCLK rising edge j, valid at most $t_{\mathrm{p(SCDO)}} = 15$ ns later [1]. With E_j the internal edge at which the PIO raises SCLK for bit j, the data are valid at the PIO input at

$$t_{\mathrm{v},j} = E_j + t_{\mathrm{loop}},$$

$$t_{\mathrm{loop}} = t_{\mathrm{po}} + 2t_{\mathrm{c}} + t_{\mathrm{p(SCDO)}} + t_{\mathrm{pi}} \qquad (6)$$

The cable delay of the unshielded wires is bounded at $t_{\mathrm{c}} \approx 0.82$–$1.12$ ns for 0.2 m ($\varepsilon_{\mathrm{eff}} = 1.5$–$2.8$). The pad delays are not specified in the RP2350 datasheet and are kept as a parameter $p = t_{\mathrm{po}} + t_{\mathrm{pi}}$. An IN instruction executed at edge E reads the synchroniser output, i.e. the pin state $n_{\mathrm{s}} = 2$ system cycles earlier [5]:

$$t_{\mathrm{s}} = E - n_{\mathrm{s}}\, T_{\mathrm{sys}} \qquad (7)$$

Consequence for the original program. Every IN executes at the edge that raises SCLK, so the sample taken at E_{j+1} lands at $E_j + (kD - n_{\mathrm{s}})T_{\mathrm{sys}} = E_j + 26.7$ ns, inside bit j's valid window and *before* bit $j + 1$ is driven (Fig. 3). Timing is safe, but the bits are misaligned: the first IN of each word samples the line before the MSB is driven, each LSB is sampled only at the first edge of the next word, and the last LSB is never sampled. Each captured word c_i therefore equals the true word d_i shifted right by one bit, with the previous word's LSB in bit 23:

$$c_i = (d_{i-1} \bmod 2) \cdot 2^{23} + \lfloor d_i/2 \rfloor \qquad (8)$$

The prediction is easy to falsify: the first word of each frame is the response to the NULL command, expected as 0x223000 (RREG(STAT_1) header 0x22) [1], but under (8) it reads 0x11…. The original firmware reported 0x111800 in every frame, i.e. 0x223000 shifted by one bit, which **confirms (8)**. Only the LSB of channel 4 is lost, far below the noise at OSR = 32.

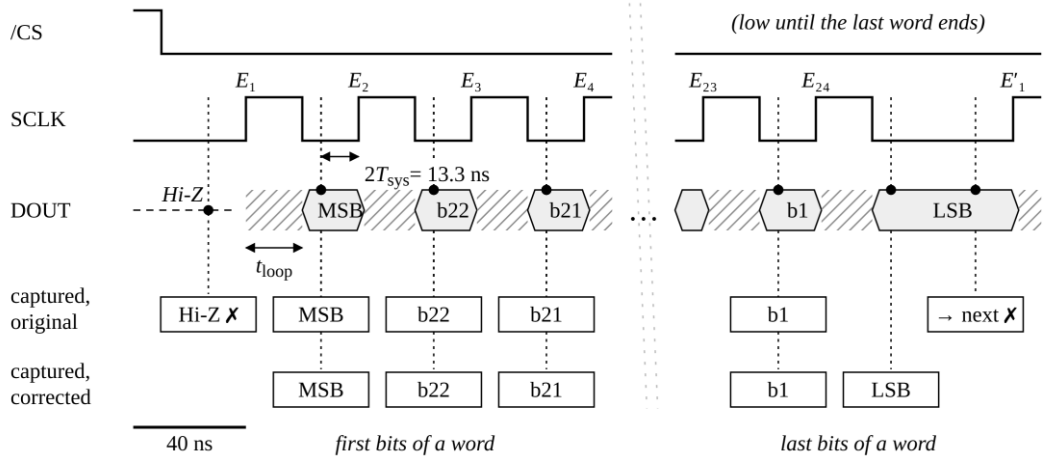


Fig. 3. *Bit sampling derived from (6)–(8): sampling instants of the original and corrected programs.*

Margins. With the corrected program (Fig. 2) the sampling instants stay at E_j + 26.7 ns, and the setup and hold margins are

$$M_{\mathrm{su}} = (kD - n_{\mathrm{s}})T_{\mathrm{sys}} - t_{\mathrm{loop}},$$
$$M_{\mathrm{h}} = n_{\mathrm{s}}T_{\mathrm{sys}} + t_{\mathrm{po}} + 2t_{\mathrm{c}} + t_{\mathrm{pi}} \qquad (9)$$

From (9), M_{su} = 9.4 ns − p and $M_{\mathrm{h}} \geq$ 15.0 ns + p at 25 MHz. Setup is the critical constraint: all margins are positive only if p < 9.4 ns. The measured $p \approx$ 6.5 ns (from the common minimum of the detection latency, 26.5 ns = 3 T_{sys} + p) leaves a guaranteed setup margin of about 2.9 ns with the datasheet maximum $t_{\mathrm{p(SCDO)}}$ = 15 ns.

Implemented configuration. To gain margin, the final firmware uses D = 1 and k = 8 system cycles per bit (4 high, 4 low): f_{SCLK} = 18.75 MHz, T_{SCLK} = 53.3 ns (33 % above the 40 ns minimum), phases of 26.7 ns and M_{su} = 22.8 ns − p (about 16 ns guaranteed, commit 2d7ec0c [12]). The frame still fits (T_{CS} = 6.52 µs, slack 0.99 µs).

EXPERIMENTAL RESULTS

A Rigol MSO5074 oscilloscope (350 MHz) with 10:1 probes at the RP2350 pins (SCLK, /CS, DOUT, /DRDY) measured three firmware versions in turn without moving the probes (Fig. 4a). Table 2 compares the measurements with the model.

Table 2. *Model predictions versus oscilloscope measurements (16.10.2026), model / measured*

Quantity	Original, 25 MHz	Bit-alignment fix, 25 MHz	Implemented, 18.75 MHz
/CS low, eq. (3)	4.97 / 4.9745 µs	5.02 / 5.0205 µs	6.52 / 6.5210 µs
Frame period, eq. (1)	7.8125 / 7.8135 µs	7.8125 / 7.8130 µs	7.8125 / 7.8135 µs
SCLK period, eq. (2)	40.0 / 40.0 ns	40.0 / 40.2 ns	53.3 / 53.3 ns
SCLK high / low	20.0 / 19.8, 20.2 ns	20.0 / 19.9, 20.3 ns	26.7 / 26.6, 26.7 ns
/DRDY↓→/CS↓, eq. (4)	20–33.3 ns + p / 36.6 ns	20–33.3 ns + p / 26.6–40.1 ns	20–26.7 ns + p / 26.5–33.6 ns

Every quantity agrees with the model within the 0.25–1 ns measurement resolution. Over about 240 triggers the detection latency spreads by 13.5 ns for D = 1.5 and 7.1 ns for D = 1, i.e. two and one system-clock cycles, as (4) predicts. Fig. 4b also shows crosstalk spikes from SCLK on /DRDY and DOUT, a risk of the loose wiring.

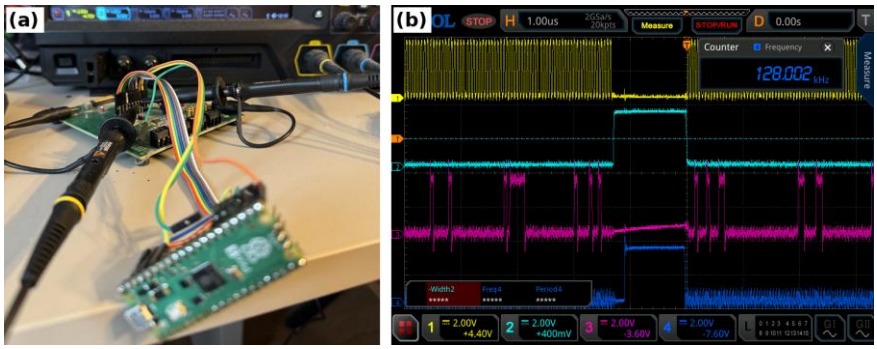


Fig. 4. *(a) Experimental set-up. (b) One frame of the 18.75 MHz version, 1 µs/div: SCLK (CH1), /CS (CH2), DOUT (CH3), /DRDY (CH4).*

Hardware verification. Each firmware step was flashed from the public repository [12]. From commit 4b416f6 on, the buffer-completion interrupt checks the RREG(STAT_1) header 0x22 and the ADC's F_DRDY flag of every frame. The ADC sets F_DRDY when the previous result was not read out before new data are ready [1], so the ADC itself referees the readout timing. The data rate is counted against the NTP-synchronised PC clock over 10 minutes (uncertainty about ±50 ppm). The original firmware returned the shifted status word 0x111800 at 128 002.6 SPS (+20.7 ppm); after the alignment fix (9830011) it reads 0x223000. With per-frame checking

(4b416f6, 25 MHz) and with the implemented $k = 8$ configuration (2d7ec0c, 18.75 MHz), 76 831 744 consecutive frames each were read without a header error or F_DRDY fault, at 128 002.2 and 128 002.7 SPS (+17.3 and +20.7 ppm), in agreement with (1). The oscilloscope counter gave f_{DRDY} = 128.002 kHz. In the final firmware (dfc757e) a GPIO marker shows an interrupt every 8.0 ms lasting 102.7 µs, i.e. about 1.3 % of one core, while the /CS frames continue at full rate.

DISCUSSION

Value of the model. The clock period, frame length and rate looked correct on the oscilloscope already in the original firmware, yet it contained a bit-alignment error that shows only in the data. The model found it, a status word confirmed it, and the model-derived program corrected it. Because the PIO is deterministic, every quantity in (2)–(9) follows from the program text and the register values; the same holds for the PRU [6], but not for bit-banging.

Limitations. The pad delay p is inferred from the detection latency, and analog performance was not evaluated (inputs unconnected). The 12.29 Mbit/s payload exceeds USB Full Speed, the only speed of the RP2350 [5] (bulk limit 1.216 MB/s [13]), so full-rate streaming needs decimation or a faster link. EnDat 2.2 work on the PRU will be reported separately.

CONCLUSIONS

Programmable I/O subsystems fill the gap between bit-banging and dedicated logic, with cycle-level determinism, no added hardware cost and firmware-level flexibility; the RP2350 PIO and the TI PRU represent a narrow I/O state machine and an I/O co-processor.

In the case study, a PIO state machine with chained DMA reads a four-channel, 24-bit, 128 kSPS ADC with one CPU interrupt per 8 ms. An analytical model from the ADC clock registers to bit-level margins was reproduced by oscilloscope measurements of three firmware versions (frame times within 1.5 ns). It showed that the original clock ran at the ADC's minimum period and revealed a one-bit misalignment invisible in the waveforms. Both were corrected on hardware: the 18.75 MHz configuration has 33 % clock margin and read 76.8 million frames without a readout fault. Deterministic programmable I/O thus allows an interface to be verified on paper before it is measured. Follow-up work will use a PCB adapter, measure the DOUT sampling margin directly and validate the analog performance.

ACKNOWLEDGMENTS*: This work is supported by the project BG05SFPR001-3.004-0018-C01 "Support for project-based PhD studies in the professional field of "Electrical Engineering, Electronics and Automation" at the Technical University - Sofia."*

REFERENCES

[1] Texas Instruments, "ADS131A0x 2- or 4-Channel, 24-Bit, 128-kSPS, Simultaneous-Sampling, Delta-Sigma ADC," Datasheet SBAS590E, Jun. 2020. https://www.ti.com/lit/gpn/ads131a04 (accessed 16.10.2026).

[2] Texas Instruments, "EnDat 2.2 System Reference Design," TIDEP0050. https://www.ti.com/tool/TIDEP0050 (accessed 16.10.2026).

[3] Microchip Technology, "Bit Banging," online documentation. https://onlinedocs.microchip.com/pr/GUID-CF2C0DBC-087A-4CE7-B9C1-5E08A928CFD9-en-US-6/GUID-52ED8983-65D6-4CC9-8A08-F6384F009DDB.html (accessed 16.10.2026).

[4] I. Lankshear, "The Economics of ASICs: At What Point Does a Custom SoC Become Viable?," Electronic Design, 15 Jul. 2019. https://www.electronicdesign.com/technologies/embedded/article/21808278/ensilica-the-economics-of-asics-at-what-point-does-a-custom-soc-become-viable (accessed 16.10.2026).

[5] Raspberry Pi Ltd, "RP2350 Datasheet," RP-008373-DS.

https://datasheets.raspberrypi.com/rp2350/rp2350-datasheet.pdf (accessed 16.10.2026).
[6] Texas Instruments, "PRU Subsystem Features Comparison," Application Note SPRAC90G, 2022. https://www.ti.com/lit/pdf/sprac90 (accessed 16.10.2026).
[7] Freescale Semiconductor (NXP), "Emulating Dual SPI Using FlexIO," Application Note AN5242, Rev. 0, Jan. 2016. https://www.nxp.com/docs/en/application-note/AN5242.pdf (accessed 16.10.2026).
[8] K. Gichev, D. Dankov, "Overview of Bidirectional GaN Switches for Next-Generation Motor Drive Applications" (in Bulgarian), in Proc. Int. Sci. Conf. UNITECH 2025, Gabrovo, Bulgaria, 20–22 Nov. 2025, Print ISSN 1313-230X, Online ISSN 3033-1404. https://unitech.tugab.bg/images/2025/dokladi/2-Electronics%20and%20Sensors/p192_s2_u176_id472.pdf (accessed 16.10.2026).
[9] Microchip Technology, "PIC16F13145 Family Microcontroller with Configurable Logic Block," Data Sheet DS40002519. https://onlinedocs.microchip.com/g/GUID-A766082F-E194-4137-9D97-0F13252F7C55 (accessed 16.10.2026).
[10] Raspberry Pi Ltd, "RP2040 Datasheet." https://datasheets.raspberrypi.com/rp2040/rp2040-datasheet.pdf (accessed 16.10.2026).
[11] M. Pauwels, J. Loeliger, "The Essentials of Enhanced Time Processing Unit," Freescale Application Note AN2353, Rev. 1, Aug. 2004. https://www.nxp.com/docs/en/user-guide/AN2353.pdf (accessed 16.10.2026).
[12] K. Gichev, "ads131_test: RP2350 PIO acquisition firmware for the ADS131A04," GitHub repository, commits d3a789e, 9830011, 4b416f6, 2d7ec0c, dfc757e. https://github.com/gicho/ads131_test (accessed 16.10.2026).
[13] USB Implementers Forum, "Universal Serial Bus Specification, Revision 2.0," 2000, Sec. 5.8.4.