Wstęp ... 13

1. Podstawy ... 15

1.1. Co oznacza ARM? ... 16

1.2. SAM7 – mikrokontrolery z rdzeniem ARM ... 17

1.3. ARM7TDMI – to warto wiedzieć ... 17

1.3.1. Architektura von Neumanna ... 17

1.3.2. Wyrównanie ... 18

1.3.3 *Little-endian* kontra *big-endian* ... 18

1.4. Jak to działa? ... 19

1.4.1. Mikrokontroler a mikroprocesor ... 19

1.4.2. Rejestry mikroprocesora ... 20

1.4.3. Rejestry ogólnego przeznaczenia i prowadzenie obliczeń ... 20

1.4.4. Licznik programu ... 21

1.4.5. Rejestr statusowy ... 21

1.4.6. Stos ... 22

1.5. Rdzeń ARM7TDMI ... 23

1.5.1. Wyjątki i przerwania ... 23

1.5.2. Poziomy uprzywilejowania ... 25

1.5.3. Tryb ARM i tryb Thumb ... 26

1.5.4. Rejestry statusowe w rdzeniu ARM ... 27

1.5.5. Pozostałe rejestry rdzenia ARM ... 28

1.5.6. Rejestry a wyjątki ... 29

1.5.7. Kontroler przerwań w SAM7 ... 31

2. Krótka powtórka z języka C ... 33

2.1. Wstęp ... 34

2.2. Założenia ... 34

2.3. Stosujmy jednolite typy zmiennych ... 35

2.4. Używanie modyfikatorów (*qualifiers*): *static*, *volatile* i *extern* ... 35

2.4.1. Zmienne *volatile* – „ulotne” ... 35

2.4.2. Zmienne *static* – „statyczne” ... 36

2.4.3. Deklaracje zmiennych i funkcji *extern* – „zewnętrzne” ... 37

2.4.4. Modyfikator *static* zastosowany do funkcji ... 37

2.5. Struktury danych ... 38

2.5.1. Krótki wstęp dla nowicjuszy ... 38

2.5.2. Co zyskujemy, czyli najprostsze zastosowania ... 39

2.5.3. Tworzenie struktur danych ... 39
2.5.4. Wyrównanie – odsłona druga ... 40
2.6. Wskaźniki ... 44
2.6.1. Co to jest wskaźnik? ... 45
2.6.2. Tworzenie wskaźników ... 45
2.6.3. Uzyskiwanie adresów zmiennych i funkcji ... 46
2.6.4. Zapis i odczyt danych za pomocą wskaźników ... 48
2.6.5. Wskaźniki do struktur ... 49
2.6.6. Wskaźniki i tablice ... 49
2.6.7. Wskaźniki a problem wyrównania ... 51
3. Strategia ... 53
3.1. Dobieramy SAM-a do naszego projektu ... 54
3.1.1. Porównanie wybranych modeli SAM7 ... 54
3.1.2. Przenośność kodu pomiędzy mikrokontrolerami SAM7 ... 56
3.1.3. Przenośność na inne mikrokontrolery firmy Atmel ... 57
3.2. Dobre nawyki ... 57
3.2.1. Podział na moduły ... 58
3.2.2. Dobry kod sam się komentuje ... 59
3.3. Testowanie i debugging ... 62
3.3.1. Sposób najprostszy – dioda LED ... 62
3.3.2. Sposób wydajny – port szeregowy UART ... 63
3.3.3. Sposób dokładny – interfejs JTAG ... 63
3.3.4. Wybór metody ... 64
3.3.5. Algorytm postępowania ... 65
3.3.6. Testowanie długoterminowe ... 65
3.4. Wybór środowiska programistycznego i narzędzi ... 66
3.4.1. Środowiska komercyjne ... 66
3.4.2. Pakiety darmowe ... 67
3.5. Nie wszystko trzeba umieć – wykorzystanie gotowych fragmentów programu ... 68
3.5.1. Biblioteki w języku C ... 69
3.5.2. Fragmenty kodu realizujące specyficzne zadania ... 69
4. Zaczynamy ... 71
4.1. Jak podłączyć SAM7? ... 72
4.1.1. Najogólniej ... 72
4.1.2. Zasilanie ... 73
4.1.3. Źródła sygnału zegarowego ... 74

4.1.4. JTAG ... 75
4.1.5. Interfejs USB ... 76
4.1.6. Podsumowanie i uwagi ... 77
4.2. Software ... 79
4.2.1. Podstawa: edytor, kompilator, programator ... 79
4.2.2. Konfiguracja portu LPT ... 82
4.2.3. Pierwsza kompilacja ... 82
4.2.4. Instalacja AT91-ISP i aktywacja bootloadera SAM-BA ... 85
4.2.4.1. Instalacja na komputerze PC ... 85
4.2.4.2. Aktywacja *bootloadera* SAM-BA w mikrokontrolerze ... 85
4.2.5. Przygotowanie SAM7X i SAM7XC do pracy i programowanie za pomocą SAM-BA ... 86
4.3. Problemy ... 87
4.3.1. Brak portu LPT ... 87
4.3.2. Nie działają narzędzia kompilacji ... 89
4.3.3. Wywoływanie poleceń ... 90
5. Omówienie pierwszego projektu ... 91
5.1. Dostęp do układów peryferyjnych ... 92
5.1.1. Do czego zmierzamy? ... 92
5.1.2. Potrzebne informacje ... 92
5.1.3. Pierwsze podejście (bezpośredni wpis do pamięci) ... 93
5.1.4. Czy można bardziej przejrzyście? (bezpośrednie adresy rejestrów) ... 94
5.1.5. Definicje bitów ... 95
5.1.6. Zadbajmy o przenośność kodu! (zastosowanie adresów bazowych) ... 97
5.2. Omówienie plików projektu *Blink_SAM7XC* ... 101
5.2.1. *main.c* ... 101
5.2.1.1. Funkcja *main* ... 101
5.2.1.2. Funkcja *init* ... 102
5.2.2. *board.h* ... 102
5.2.3. *Makefile* ... 104
5.2.3.1. Wybór typu mikrokontrolera ... 104
5.2.3.2. Dodawanie nowych plików projektu ... 105
5.2.3.3. Zmiana poziomu optymalizacji ... 105
5.2.4. *Linker scripts* – pliki *.ld* ... 106
5.2.4.1. Omówienie najistotniejszego fragmentu ... 106
5.2.4.2. Podział pamięci na sekcje ... 107
5.2.4.3. Nazwy plików *.ld* ... 107
5.2.5. Pliki startowe – ogólnie ... 107

6. Kontroler PIO czyli port uniwersalnych wejść/wyjść cyfrowych 109

6.1. Wstęp 110

6.2. Włączenie sygnału zegarowego kontrolera PIO 110

6.3. Konfiguracja jako wyjście: miganie diodą LED 111

6.3.1. Pierwszy sposób generowania sygnału wyjściowego (SODR, CODR) 111

6.3.2. Drugi sposób sterowania sygnału wyjściowego (ODSR) 112

6.3.3. Przykład do skompilowania 114

6.3.4. Rejestry komplementarne 115

6.4. Praca jako wejście cyfrowe: odczyt stanu przycisków 116

6.4.1. Sposób najprostszy 116

6.4.2. Program demonstracyjny 117

6.4.3. Dalsze usprawnienia odczytu 117

6.5. Obsługa popularnych wyświetlaczy LCD 2×16 znaków 118

6.5.1. Hardware 118

6.5.2. Sposób sterowania wyprowadzeniami PIO 119

6.5.3. Realizacja praktyczna 120

6.5.4. Programowy interfejs wyświetlacza 122

6.6. Interfejs 1-Wire 123

6.6.1. Hardware 123

6.6.2. Konfiguracja PIO 124

6.6.3. Funkcje wyższego poziomu (z pliku *one_wire.c*) 126

6.6.4. Funkcje obsługi termometru DS18B20 127

6.7. Przerwanie od kontrolera PIO 127

6.7.1. Wstęp dla niewtajemniczonych 128

6.7.2. Przykładowy program 129

6.7.3. Konfiguracja kontrolera przerwań AIC 129

6.7.4. Konfiguracja układu PIO 130

6.7.5. Funkcja obsługi przerwania PIO 131

6.7.6. Dla dociekliwych: jak to jest naprawdę? 133

7. Najprostsza komunikacja szeregowa przez DBGU oraz elementy interfejsu USART 135

7.1. Wstęp 136

7.1.1. Co to jest i do czego służy jednostka DBGU? 136

7.1.2. DBGU a typowy port szeregowy 136

7.2. Sposób podłączenia 136

7.2.1. Hardware 137

7.2.2. Brak portu COM 138

7.3. Konfiguracja komputera PC ... 139
7.4. Najprostsze funkcje obsługi portu szeregowego układu DBGU ... 140
7.4.1. Przejęcie kontroli nad wyprowadzeniami przez układ peryferyjny ... 140
7.4.2. Zastosowanie definicji bitów kontrolera PIO z plików nagłówkowych Atmela ... 142
7.4.3. Konfiguracja rejestrów portu DBGU ... 143
7.4.4. Ostateczna postać funkcji inicjalizującej ... 146
7.4.5. Wysyłanie znaku przez port szeregowy ... 147
7.4.6. Wysyłanie znaku dla bardziej dociekliwych ... 148
7.4.7. Odbieranie danych ... 149
7.5. Funkcje wyższego poziomu ... 150
7.5.1. Pisanie tekstów ... 150
7.5.2. Wyświetlanie liczb ... 150
7.5.3. Wyświetlanie obszarów pamięci na terminalu ... 151
7.5.4. Odczyt danych wprowadzonych przez użytkownika ... 152
7.5.5. Funkcje dostępu do przestrzeni adresowej ... 152
7.5.6. Przykład zastosowania funkcji *debug_menu* ... 153
7.6. Port szeregowy w DBGU a prawdziwy USART ... 155
7.6.1. Modyfikacja wywołania funkcji inicjalizującej ... 155
7.6.2. Uzupełnienie jednego pola bitowego w *Mode Register* ... 156
7.6.3. Jeszcze tylko przełączenie wtyczki ... 156
7.6.4. Podsumowanie ... 157
8. Zaawansowane sposoby komunikacji szeregowej przez USART ... 159
8.1. Wstęp ... 160
8.2. Zastosowanie systemu przerwań do modułu USART ... 160
8.2.1. Program przykładowy ... 160
8.2.2. Inicjalizacja portu szeregowego do pracy z przerwaniami ... 161
8.2.3. Funkcja obsługi przerwania od układu USART ... 164
8.2.4. Wysyłanie danych w funkcji obsługi przerwania ... 165
8.3. Bezpośredni dostęp do pamięci, czyli DMA ... 168
8.3.1. Co to jest DMA oraz po co się to stosuje? ... 168
8.3.2. Implementacja DMA w mikrokontrolerach SAM7 ... 168
8.3.3. Najprostszy przykład transferów DMA ... 170
8.3.4. Funkcja inicjalizująca USART ... 172
8.3.5. Najprostsza realizacja wysyłania danych przez DMA ... 172
8.3.6. O tych rejestrach także warto wiedzieć (wysyłanie) ... 173
8.3.7. Najprostsza realizacja odbioru danych przez DMA ... 175
8.3.8. O tym także warto wiedzieć (odbiór) ... 176

8.4. Połączenie transferów DMA i układu przerwań ... 176
8.4.1. Program przykładowy i cel działań ... 177
8.4.2. Komunikacja szeregowa z *handshakingiem* – jak się to podłącza? ... 180
8.4.3. Działanie sygnałów RTS i CTS ... 181
8.4.4. Timeout w odbiorniku – sposób na zróżnicowaną liczbę danych wejściowych ... 181
8.4.5. Funkcja inicjalizująca ... 182
8.4.6. Odbiór danych ... 186
8.4.7. Wysyłanie danych ... 190
8.5. Podsumowanie ... 191
8.5.1. Gdzie nie sprawdzi się DMA i przerwania ... 191
8.5.2. Nie tylko USART ... 191
9. SPI – działanie i zastosowania ... 193
9.1. Wstęp ... 194
9.1.1. Jak to działa, do czego służy i czym się charakteryzuje? ... 194
9.2. Hardware ... 194
9.2.1. Połączenie fizyczne ... 195
9.2.2. Sygnały elektryczne ... 196
9.3. Obsługa interfejsu SPI ... 197
9.3.1. Kilka słów o programie testowym ... 197
9.3.2. Inicjalizacja SPI: najpierw ogólna ... 197
9.3.3. Inicjalizacja poszczególnych kanałów SPI ... 200
9.3.4. Przebieg transmisji danych ... 206
9.3.5. Funkcja wysyłająca i odbierająca pojedynczy znak ... 209
9.3.6. Funkcja transmitująca dane w trybie DMA ... 210
9.3.7. Praktyczne uwagi na temat transferów danych ... 214
9.4. Obsługa wyświetlacza graficznego do telefonu Nokia 3310 ... 216
9.4.1. Sposób podłączenia ... 216
9.4.2. Program przykładowy ... 217
9.4.2. Niskopoziomowa inicjalizacja wyświetlacza ... 217
9.4.3. Jak to działa, czyli organizacja przepływu danych ... 219
9.4.4. Omówienie najważniejszych funkcji obsługi wyświetlacza ... 220
9.5. Obsługa karty SD ... 221
9.5.1. Podłączenie karty do mikrokontrolera ... 221
9.5.2. Program przykładowy zamiast teorii ... 222
9.5.3. Krótko o inicjalizacji interfejsu SPI dla karty ... 224
9.5.4. Funkcje obsługi karty SD ... 225
9.6. Podobny interfejs – SSC ... 226

10. System plików FAT 227

10.1. Wstęp 228

10.1.1. Co zyskamy? 228

10.1.2. W jaki sposób zaimplementujemy obsługę systemu plików? 228

10.1.3. O bibliotece FatFs 228

10.2. Zastosowanie FatFs w projekcie 229

10.2.1. Pliki, moduły i funkcje... czyli którędy płyną dane 229

10.2.2. Dołączenie funkcji obsługi karty SD do biblioteki FatFs 229

10.3. Obsługa biblioteki FatFs – minimum teorii 231

10.3.1. Program przykładowy FAT_Test 232

10.3.2. Idea działania 232

10.3.3. Otwieranie pliku 233

10.3.4. Odczyt danych z pliku 235

10.3.5. *File pointer* 237

10.3.6. Zapis danych do pliku 238

10.3.7. Inne funkcje FatFs 239

10.4. Funkcje narzędziowe *fs_tools.c* 240

10.5. Projekt przykładowy: rejestrator temperatury 241

10.6. Projekt przykładowy: prawie jak animacja 242

10.6.1. Przygotowania 242

10.6.2. Działanie programu przykładowego 242

10.6.3. Format BMP w praktyce 242

10.6.4. Wyświetlanie obrazu 245

10.6.5. Inne zastosowania 246

11. Przetwornik ADC 247

11.1. Przyspieszony kurs dla początkujących 248

11.2. Najprostszy przykład z ADC w SAM7 249

11.2.1. Sprzęt 249

11.2.2. Program przykładowy 250

11.2.3. Odczyt ADC w najprostszym wydaniu 251

11.2.4. Konfiguracja *timingu* ADC 253

11.3. Próbkowanie sygnału 255

11.3.1. Przyspieszony kurs: jak szybko próbkować? 255

11.3.2. Program przykładowy 257

11.3.3. Konfiguracja ADC 258

11.3.4. Inicjalizacja układu *Timer Counter* dla ADC 259

11.3.5. Funkcja obsługi przerwania 262

11.4. Dyktafon cyfrowy – połączenie ADC, DMA i elementów techniki analogowej ... 262
11.4.1. Hardware ... 263
11.4.2. Projekt *Peak_Level_Meter* – tester mikrofonu ... 265
11.4.3. Obsługa dyktafonu ... 265
11.4.4. Co się dzieje w programie? ... 266
11.4.5. Inicjalizacja ADC do pracy w trybie DMA ... 267
11.4.6. Podwójne buforowanie w praktyce – funkcja *adcDmaBufferSwap* ... 268
11.4.7. Format pliku WAVE – minimum teorii ... 269
11.4.8. Zapis dźwięku do formatu WAVE w projekcie dyktafonu ... 273

12. Generator PWM ... 277

12.1. Zło konieczne, czyli trochę teorii ... 278
12.1.1. Co to jest PWM? ... 278
12.1.2. Przykład zastosowania: regulacja jasności świecenia diod LED ... 278
12.1.3. Składowa stała sygnału PWM ... 279
12.1.4. Jak uzyskać wartość składowej stałej z przebiegu PWM? ... 280
12.2. Kontroler PWM w SAM7 ... 281
12.2.1. Typowy generator PWM ... 281
12.2.3. Dostęp do rejestrów kontrolera PWM ... 283
12.3. Zastosowanie PWM do regulacji jasności podświetlenia wyświetlacza LCD ... 284
12.3.1. Hardware ... 285
12.3.2. Działanie programu przykładowego ... 285
12.3.2. Inicjalizacja kontrolera PWM ... 285
12.3.4. Ustawianie współczynnika wypełnienia ... 287
12.4. Odtwarzacz plików dźwiękowych ... 288
12.4.1. Obsługa i działanie odtwarzacza ... 288
12.4.2. Hardware – znowu analogówka ... 289
12.4.3. Inicjalizacja kontrolera PWM i układu *Timer Counter* ... 290
12.4.4. Odczyt plików WAVE ... 293
12.5. Podsumowanie i pomysły ... 293

13. USB od podstaw ... 295

13.1. Wstęp ... 296
13.2. USB – jak zacząć? ... 296
13.2.1. Wiadomości ogólne ... 296
13.2.2. Od bitu do pakietu ... 297
13.2.3. Endpoint i *pipe* ... 299

13.3. Transfery danych ... 301
13.3.1. *Control Transfers* ... 301
13.3.2. *Bulk Transfers* ... 305
13.3.3. *Isochronous Transfers* ... 307
13.3.4. *Interrupt Transfers* ... 308
13.4. Programowa realizacja transferów danych ... 309
13.4.1. *Ping-pong*, czyli *Dual-Bank* ... 309
13.4.2. Obsługa transferów kontrolnych ... 309
13.4.2.1. *Setup Stage* ... 309
13.4.2.2. *Data Stage* i *Status Stage* ... 312
13.4.3. Pakiety ZLP i STALL w transferach kontrolnych ... 315
13.4.3.1. Wysyłanie pakietu ZLP ... 315
13.4.3.2. Wysyłanie pakietu STALL ... 315
13.4.4. Transfery Data OUT do endpointów z atrybutem *Dual-Bank* ... 316
13.4.5. Transfery Data IN z endpointów z atrybutem *Dual-Bank* ... 318
13.5. Treść transferów kontrolnych ... 320
13.5.1. Pakiet SETUP pod lupą ... 321
13.5.2. Czego żąda host i co z tym zrobić? ... 322
13.5.2.1. Przykłady żądań dla wszystkich klas ... 322
13.5.2.2. Przykłady żądań specyficznych dla klasy CDC ... 325
13.6. Deskryptory ... 326
13.6.1. Model urządzenia USB ... 327
13.6.2. Ogólnie o deskryptorach ... 327
13.6.3. *Device Descriptor* ... 328
13.6.4. *Configuration Descriptor* ... 329
13.6.5. *Interface Descriptor* ... 330
13.6.6. *Endpoint Descriptor* ... 330
13.7. Projekt przykładowy – przejściówka USB-RS232 ... 331
13.7.1. Funkcjonalność ... 332
13.7.2. Program główny ... 332
13.7.3. Funkcje odpowiedzialne za transfery USB ... 333
13.7.4. Zastosowanie kodu z przykładu *USB2Serial* ... 335
13.8. Projekt przykładowy – czytnik kart SD z interfejsem USB ... 336
13.8.1. Funkcjonalność ... 336
13.8.2. Zarys działania ... 336
13.8.3. Obsługa interfejsu USB ... 338
13.8.4. Jak przenieść kod obsługi MSD do własnego projektu? ... 339
13.8.5. Program przykładowy – kolejna odsłona rejestratora temperatury ... 341

14. Deser: szyfrowanie danych ... 343

14.1. Wstęp ... 344

14.2. AES w przykładzie ... 344

14.2.1. Obsługa programu przykładowego ... 344

14.2.2. Pokaz pierwszy: szyfrowanie i deszyfrowanie bufora z danymi ... 345

14.2.3. Pokaz drugi: różne klucze ... 345

14.2.4. Pokaz trzeci: same zera ... 346

14.3. Obsługa modułu kryptograficznego AES ... 346

14.3.1. Szyfrowanie i deszyfrowanie ... 347

14.3.2. Ustawianie klucza ... 348

14.4. Co jeszcze warto wiedzieć o module AES? ... 348

14.4.1. Kontrola pracy modułu AES ... 349

14.4.2. Wybór trybu szyfrowania ... 350

14.4.3. Zabezpieczenia ... 353

14.5. Przykład praktyczny – szyfrator kart SD ... 353

14.5.1. Działanie programu ... 353

14.5.2. Realizacja szyfrowania ... 354

14.6. Co warto wiedzieć oraz często zadawane pytania ... 355

14.6.1. Zastosowania szyfrowania danych ... 355

14.6.2. Gdzie jest klucz? ... 356

Dodatek. ZL28ARM – zestaw uruchomieniowy z mikrokontrolerem AT91SAM7XC ... 357

Literatura ... 359