Comprehensive Embedded AI & Signal Processing Notes

Table of Contents

---

1. Machine Learning Hardware

1.1 ALU vs MAC vs Accumulator

ALU (Arithmetic Logic Unit)

- **Function**: Performs basic arithmetic (addition, subtraction) and logical operations (AND, OR, XOR)
- **Operations**: Single-cycle operations
- **Use Case**: General-purpose computing, control flow
- **Example**:

A = 5 + 3 = 8
B = 10 - 2 = 8

C = A AND B = 8 AND 8 = 8

MAC (Multiply-Accumulate Unit)

- **Function**: Performs multiplication followed by addition in a single operation
- **Operations**: `Result = (A × B) + C`
- **Use Case**: Neural networks, DSP, matrix operations
- **Efficiency**: Critical for ML - most NN operations are dot products

Example Calculation:

```
Neural Network Weight Calculation:
Weights: [0.5, 0.3, 0.2]
Inputs:  [1.0, 2.0, 3.0]

MAC Operation:
Result = (0.5 × 1.0) + (0.3 × 2.0) + (0.2 × 3.0)
       = 0.5 + 0.6 + 0.6
       = 1.7

Traditional ALU would need:
Step 1: 0.5 × 1.0 = 0.5
Step 2: 0.3 × 2.0 = 0.6
Step 3: 0.2 × 3.0 = 0.6
Step 4: 0.5 + 0.6 = 1.1
Step 5: 1.1 + 0.6 = 1.7
```

(5 operations vs 1 MAC operation)

Accumulator

- **Function**: Special register that stores intermediate results
- **Role**: Holds the sum in MAC operations
- **Width**: Determines precision (8-bit, 16-bit, 32-bit, etc.)

1.2 Accumulator Width Impact

Accumulator Width	Range	ML Impact
8-bit	0 to 255 (unsigned)	Limited precision, overflow risk, suitable for simple models
16-bit	0 to 65,535	Better for quantized networks
32-bit	0 to 4,294,967,295	High precision, prevents overflow in deep networks

Example:

```
16-bit Accumulator:
Max value = 65,535

If we have 1000 MAC operations:
Average value per operation = 65,535 / 1000 = 65.5
This limits weight × input product to ~65

32-bit Accumulator:
```

Max value = 4,294,967,295
Average per 1000 ops = 4,294,967

Much more headroom for complex calculations

1.3 Ethos-U55 MAC Engine Configurations

Configuration	MAC Units	Performance	Use Case
32 MAC	32	Lowest power	Keyword spotting, simple inference
64 MAC	64	Balanced	Face detection, gesture recognition
128 MAC	128	High performance	Object detection
256 MAC	256	Maximum performance	Real-time video processing

---

2. ARM Cortex-M55 Processor

2.1 Key Features

Cortex-M55 Specifications:

- **First CPU with ARM Helium Technology** (M-profile Vector Extension - MVE)
- **Performance Gains:**
  - Up to **15× ML performance** vs previous Cortex-M generations
  - Up to **5× DSP performance**
- **Vector Processing**: SIMD (Single Instruction Multiple Data)
- **TrustZone Support**: Hardware-level security
- **Power Efficiency**: Designed for battery-powered devices

2.2 Helium Technology

What is Helium?

- Vector processing extension for Cortex-M
- Processes multiple data elements in parallel
- Optimized for ML and DSP workloads

Example:

Traditional Processing (without Helium):
Add 4 numbers: [1, 2, 3, 4] + [5, 6, 7, 8]
Cycle 1: 1 + 5 = 6

Cycle 2: 2 + 6 = 8
Cycle 3: 3 + 7 = 10
Cycle 4: 4 + 8 = 12
Total: 4 cycles

With Helium (SIMD):
Cycle 1: [1,2,3,4] + [5,6,7,8] = [6,8,10,12]

Total: 1 cycle (4× faster!)

---

3. Ethos-U MicroNPUs

3.1 Ethos-U55 (First Generation)

Architecture:

```
┌─────────────────────────────────────┐
│  Cortex-M Processor                 │
└─────────────────────────────────────┘
        │
┌─────────────────────────────────────┐
│  Ethos-U55 MicroNPU                 │
│ ┌─────────────────────────────┐     │
│ │ Control Unit                │     │
│ └─────────────────────────────┘     │
│ ┌─────────────────────────────┐     │
│ │ MAC Engine (32/64/128/256)  │     │
│ └─────────────────────────────┘     │
│ ┌─────────────────────────────┐     │
│ │ Weight Decoder              │     │
│ └─────────────────────────────┘     │
│ ┌─────────────────────────────┐     │
│ │ Local Memory                │     │
│ └─────────────────────────────┘     │
└─────────────────────────────────────┘
     │          │
  System SRAM  System Flash
```

Key Features:

- **MAC Options**: 32, 64, 128, or 256 units
- **Weight Compression**: On-the-fly decompression (up to 90% SRAM reduction)
- **Compatible Processors**: Cortex-M55, M33, M7, M4
- **Memory**: Works with SRAM and Flash

3.2 Ethos-U65 (Second Generation)

Improvements over U55:

- **MAC Options**: 256 or 512 units (2× more powerful)
- **DRAM Support**: Better for larger models
- **Hybrid Systems**: Can work in A-class systems (Cortex-A processors)

3.3 Suitable Applications

Ethos-U55 Applications (From Exam Q2)

1. **BLE Sense Nano (Cortex-M55 + Ethos-U55)**
   - **Object Classification**: Identify objects in images
   - **Why suitable?**:
     - Low power for battery operation
     - On-device processing (privacy)
     - Fast inference (<100ms)
2. **Real-world Examples:**
   - **Keyword Spotting**: "Hey Google", "Alexa"
   - **Predictive Maintenance**: Vibration analysis in motors
   - **Gesture Recognition**: Smart home controls
   - **Face Unlock**: Smartphone security

Application Mapping by Performance

```
Performance Scale:
├─ Cortex-M4 (Baseline)
│   └─ Sensor fusion, basic keyword detection
│
├─ Cortex-M55 (15× improvement)
│   └─ Speech recognition, anomaly detection
│
├─ Cortex-M55 + Ethos-U55 (25× improvement)
│   └─ Object detection, gesture recognition
│
└─ Cortex-A + Ethos-N (1000× improvement)
    └─ Real-time video analysis, complex vision
```

---

4. CMSIS-NN

4.1 What is CMSIS-NN?

CMSIS (Cortex Microcontroller Software Interface Standard)

- Software library for Cortex-M processors

- **CMSIS-NN**: Neural Network optimized kernels
- Provides efficient implementations of common NN operations

4.2 Performance Comparison

From Document - Wav2Letter Performance:

```
Baseline (CM4 Reference):       1×
CM4 with CMSIS:               3.5×
Cortex-M55 with CMSIS:         11×
Cortex-M55 + Ethos-U55:        25×

Total Improvement: ~1000× from baseline to U55!
```

4.3 TensorFlow Lite Micro Integration

Workflow:

```
1. Train Model (TensorFlow)
2. Convert to .tflite file
3. Optimize with Vela Compiler
4. Deploy to device with TFLite Micro
5. Run inference using:
  - Reference kernels (slow)
  - CMSIS-NN kernels (fast)
  - Ethos-U microNPU (fastest)
```

---

5. ADC and Quantization

5.1 N-bit ADC Resolution

Formula:

Number of Quantization Levels = 2^N

Voltage Resolution = (V_max - V_min) / (2^N)

5.2 10-bit ADC Example (Step-by-Step)

Given:

- Reference Voltage: 5V
- ADC: 10-bit
- Input Signal: 3.2V

Step 1: Calculate Quantization Levels

Levels = 2^10 = 1024 levels

Step 2: Calculate Resolution (LSB)

Resolution = V_ref / Levels
= 5V / 1024

= 0.00488V (4.88mV per step)

Step 3: Convert Analog to Digital

Digital Value = (V_input / V_ref) × (2^N - 1)
= (3.2V / 5V) × 1023
= 0.64 × 1023
= 654.72 ≈ 655 (rounded)

Binary: 655 = 1010001111 (10 bits)

Step 4: Quantization Error

Actual Voltage = (655 / 1023) × 5V = 3.201V

Quantization Error = 3.201V - 3.2V = 0.001V = 1mV

5.3 Different ADC Resolutions

ADC Bits	Levels	Resolution (5V)	Accuracy
8-bit	256	19.5mV	±9.75mV
10-bit	1024	4.88mV	±2.44mV
12-bit	4096	1.22mV	±0.61mV
16-bit	65,536	76.3μV	±38.1μV

5.4 Linear Quantization

Process:

1. Divide voltage range into equal intervals
2. Map each interval to a digital code
3. Round to nearest level

Example: 3-bit ADC (8 levels), 0-8V range

Level 0: 0.0 - 1.0V → 000

Level 1: 1.0 - 2.0V → 001
Level 2: 2.0 - 3.0V → 010
Level 3: 3.0 - 4.0V → 011
Level 4: 4.0 - 5.0V → 100
Level 5: 5.0 - 6.0V → 101
Level 6: 6.0 - 7.0V → 110
Level 7: 7.0 - 8.0V → 111

Input: 5.7V → Level 5 → Binary: 101

5.5 Brain Signal Interfacing (EEG)

From Exam Q1: EEG Signal Processing

Given:

- EEG Signal: Up to ~70 Hz
- Application: Brain-Computer Interface

Step 1: Apply Nyquist Criterion

Minimum Sampling Rate = 2 × f_max
= 2 × 70 Hz
= 140 Hz

Practical Sampling Rate: 256 Hz or 512 Hz

(Power of 2 for FFT efficiency)

Step 2: Choose ADC Resolution

EEG Signal Amplitude: ~10μV to 100μV (typical)
Noise Level: ~1μV

Required SNR: 40-60 dB

For 12-bit ADC:
Dynamic Range = 20 × log10(2^12) = 72 dB ✓

Recommended: 12-bit or 16-bit ADC

Step 3: Calculate Data Rate (16 channels)

Data Rate = Sampling Rate × Bits × Channels
= 256 Hz × 16 bits × 16 channels
= 65,536 bits/second

= 8.192 KB/second

5.6 Non-Linear Quantization

Why Non-Linear?

- Human perception is logarithmic (hearing, vision)
- More resolution for small signals, less for large

μ-law Encoding (Audio):

Formula: $F(x) = \text{sign}(x) \times \ln(1 + \mu|x|) / \ln(1 + \mu)$

Where $\mu = 255$ (North America) or 256 (International)

Example:
Input: 0.1 (normalized)
Output: $\text{sign}(0.1) \times \ln(1 + 255 \times 0.1) / \ln(256)$
$= \ln(26.5) / \ln(256)$
$= 0.587$

This gives more bits to quiet sounds!

A-law Encoding (Used in Europe):

- Similar to μ-law but different curve
- Better for small amplitude signals

---

6. Sampling Theory

6.1 Nyquist Criterion

Theorem:

$f_{sampling} \geq 2 \times f_{max}$

Where f_{max} is the highest frequency in the signal

Why 2×?

- Need at least 2 samples per cycle to reconstruct signal
- Prevents aliasing (frequency folding)

6.2 Practical Example

Audio Signal: 20 Hz - 20 kHz

Minimum Sampling Rate = 2 × 20,000 Hz = 40 kHz

Actual CD Quality = 44.1 kHz (10% overhead)

EEG Signal: 0.5 Hz - 70 Hz

Minimum = 2 × 70 Hz = 140 Hz

Practical = 256 Hz (allows filtering)

6.3 Aliasing Effect

The "Wagon Wheel Effect"

- Wheel spins forward but appears to move backward
- Happens when sampling rate < 2 × rotation frequency

Example:

Wheel rotating at 10 Hz (10 revolutions/second)
Camera at 15 FPS

15 FPS < 2 × 10 Hz = 20 Hz (violates Nyquist!)

Result: Appears to rotate backward

Solution: Film at ≥20 FPS

Mathematical Explanation:

True frequency: f = 10 Hz
Sampling: fs = 15 Hz

Aliased frequency = |f - fs| = |10 - 15| = 5 Hz

Appears as 5 Hz backward rotation!

---

7. Signal Processing

7.1 Fourier Transform

Purpose: Convert time-domain signal to frequency-domain

Time Domain → Frequency Domain

Signal: x(t) = A×sin(2πft)

Fourier Transform gives:
- Frequency components
- Amplitude of each frequency
- Phase of each frequency

Example:

Mixed Signal: $x(t) = 2\times\sin(2\pi\times50t) + 1\times\sin(2\pi\times120t)$

Time Domain: Complex waveform
Frequency Domain:
- Peak at 50 Hz (amplitude = 2)
- Peak at 120 Hz (amplitude = 1)

7.2 FFT (Fast Fourier Transform)

Efficiency:

- DFT: $O(N^2)$ operations
- FFT: $O(N \log N)$ operations

For N = 1024 samples:

DFT: $1{,}024^2 = 1{,}048{,}576$ operations
FFT: $1024 \times \log2(1024) = 1024 \times 10 = 10{,}240$ operations

FFT is 102× faster!

Practical Implementation:

Input: 512 samples at 256 Hz
FFT Size: 512 (must be power of 2)

Frequency Resolution = Sampling Rate / FFT Size
= 256 Hz / 512
= 0.5 Hz per bin

Output: 256 frequency bins (0-128 Hz)

7.3 Determining Signal Frequencies

Step-by-Step Process:

Given Signal:

$x(t) = 3\times\sin(2\pi\times10t) + 2\times\sin(2\pi\times25t) + 0.5\times\sin(2\pi\times50t)$

Step 1: Identify Components

- Component 1: f_1 = 10 Hz, A_1 = 3
- Component 2: f_2 = 25 Hz, A_2 = 2
- Component 3: f_3 = 50 Hz, A_3 = 0.5

Step 2: Determine f_max

f_max = maximum frequency = 50 Hz

Step 3: Required Sampling Rate

fs = 2 × f_max = 2 × 50 = 100 Hz minimum

Practical: 128 Hz or 256 Hz

---

8. Filters

8.1 Notch Filter

Purpose: Remove specific frequency (noise)

Common Applications:

- Remove 50/60 Hz power line noise
- Remove specific interference

Design:

Center Frequency (f_0): Frequency to remove
Bandwidth (BW): Range around f_0
Quality Factor (Q): f_0 / BW

Example:
Remove 60 Hz with 2 Hz bandwidth

Q = 60 / 2 = 30 (narrow notch)

Frequency Response:

```
Gain (dB)
  0 |    /\        /\
-10 |   /  \      /  \
-20 |  /    \    /    \
-40 | /      \____/      \
    └──────────────────────────── Frequency (Hz)
     0   50  60  70     100
              ↑
```

Notch at 60 Hz

8.2 Other Common Filters

Low-Pass Filter:

- Passes frequencies below cutoff
- Use: Remove high-frequency noise

High-Pass Filter:

- Passes frequencies above cutoff
- Use: Remove DC offset, low-frequency drift

Band-Pass Filter:

- Passes frequencies in a range
- Use: Isolate specific frequency band

---

9. Audio Processing

9.1 Mel Frequency Scale

Why Mel Scale?

- Human hearing is logarithmic, not linear
- We perceive pitch differences logarithmically

Conversion:

```
Mel(f) = 2595 × log10(1 + f/700)

Examples:
f = 0 Hz → Mel = 0
f = 1000 Hz → Mel = 1127
f = 2000 Hz → Mel = 1842
f = 4000 Hz → Mel = 2555
```

Mel Filter Bank:

```
Frequency (Hz)
0     1000    2000    4000    8000
|————|——————|——————|——————|
 Wide       Narrow    Narrow

(Linear spacing in Mel scale)
```

9.2 MFCC (Mel-Frequency Cepstral Coefficients)

Purpose: Feature extraction for speech recognition

Process:

1. Frame signal (20-40ms windows)
2. Apply FFT → Frequency spectrum
3. Apply Mel filter bank → Mel spectrum
4. Take logarithm → Log Mel spectrum
5. Apply DCT → MFCC features

Step-by-Step Example:

Given: Speech signal at 16 kHz

Step 1: Frame
Window size = 25ms = 0.025s
Samples per frame = 16,000 × 0.025 = 400 samples

Step 2: FFT
FFT size = 512 (next power of 2)
Output: 256 frequency bins (0-8 kHz)

Step 3: Mel Filter Bank (40 filters)
Output: 40 Mel-scaled values

Step 4: Log
Output: 40 log-scaled values

Step 5: DCT
Keep first 13 coefficients (MFCC 1-13)

Final Output: 13 features per frame

Why MFCCs are Powerful:

- Compact representation (13 numbers instead of 400)
- Captures timbral characteristics of speech
- Robust to noise

---

10. ATmega328P Microcontroller

10.1 Overview

Specifications:

- **Architecture:** 8-bit AVR RISC
- **Clock:** Up to 20 MHz
- **Flash:** 32 KB
- **RAM:** 2 KB
- **EEPROM:** 1 KB
- **Package:** DIP-28, TQFP-32, QFN-32

10.2 Pin Configuration (DIP-28 Package)

```
       ┌─────────────────────┐
PC6 ─┤ 1 RESET  VCC ├28─ VCC (5V)
PD0 ─┤ 2  RXD   GND ├27─ GND
PD1 ─┤ 3  TXD  XTAL ├26─ PC5 (ADC5/SCL)
PD2 ─┤ 4       XTAL ├25─ PC4 (ADC4/SDA)
PD3 ─┤ 5        PC3 ├24─ PC3 (ADC3)
PD4 ─┤ 6        PC2 ├23─ PC2 (ADC2)
VCC ─┤ 7        PC1 ├22─ PC1 (ADC1)
GND ─┤ 8        PC0 ├21─ PC0 (ADC0)
PB6 ─┤ 9        GND ├20─ GND
PB7 ─┤ 10      AREF ├19─ AREF
PD5 ─┤ 11       PB5 ├18─ PB5 (SCK)
PD6 ─┤ 12       PB4 ├17─ PB4 (MISO)
PD7 ─┤ 13       PB3 ├16─ PB3 (MOSI)
PB0 ─┤ 14       PB2 ├15─ PB2
       └─────────────────────┘
```

10.3 Pin Functions

Port B (PB0-PB7)

- **PB0-PB5**: Digital I/O
- **PB3-PB5**: SPI (MOSI, MISO, SCK)
- **PB6-PB7**: Crystal oscillator

Port C (PC0-PC5)

- **PC0-PC5**: Digital I/O + ADC (6 channels)
- **PC4-PC5**: I2C (SDA, SCL)
- **PC6**: RESET pin

Port D (PD0-PD7)

- **PD0-PD1**: UART (RX, TX)
- **PD2-PD3**: External interrupts (INT0, INT1)
- **PD3, PD5, PD6**: PWM outputs
- **PD4, PD7**: Timer inputs

10.4 RESET Pin Function

PC6/RESET Pin:

- **Active Low**: Pulling to ground resets MCU
- **Internal Pull-up**: Normally HIGH
- **Reset Sources:**
    1. Power-on Reset
    2. External Reset (button)
    3. Watchdog Reset
    4. Brown-out Reset

Reset Circuit:

```
VCC (5V)
  |
  └──┤ 10kΩ ├──┬─── RESET Pin
         |
      [Button]
         |
        GND
```

10.5 Clock System

Internal Oscillator:

- 8 MHz factory calibrated
- Can be prescaled to lower frequencies

External Crystal:

- Up to 20 MHz
- More accurate for timing-critical applications

Clock Frequency Calculations:

```
Timer Frequency = Clock / Prescaler

Example: 16 MHz clock with prescaler 64
Timer Freq = 16,000,000 / 64 = 250,000 Hz

Time per tick = 1 / 250,000 = 4 μs
```

10.6 Quad Package (QFN/TQFP)

Advantages:

- Smaller footprint

- Better thermal performance
- More pins (32 vs 28)

Additional Pins in 32-pin package:

- More power/ground pins
- Additional Port C pins
- Better noise immunity

---

11. Confusion Matrix Calculations

11.1 Understanding the Matrix

From Exam Q3:

```
Confusion Matrix (Motion Detection):
              Predicted
              Circle | Idle | Left-Right | Up-Down
Actual  Circle     205  |  10  |     1      |  46
       Idle        6  | 199  |     0      |  32
       Left-Right   9  |  17  |   223      |  34
       Up-Down     21  |   8  |     3      | 186
```

11.2 Calculate Metrics for Each Class

Circle Class:

True Positive (TP): 205 (correctly classified as Circle)

False Positive (FP):

```
FP = Predicted Circle but actually other classes
   = 6 (Idle) + 9 (Left-Right) + 21 (Up-Down)
   = 36
```

False Negative (FN):

```
FN = Actually Circle but predicted as other
   = 10 (Idle) + 1 (Left-Right) + 46 (Up-Down)
   = 57
```

True Negative (TN):

```
TN = All other correct classifications
   = 199 + 0 + 32 + 223 + 34 + 3 + 186
```

= 677

Per-Class Accuracy:

Accuracy = (TP + TN) / (TP + TN + FP + FN)
= (205 + 677) / (205 + 677 + 36 + 57)
= 882 / 975
= 0.9046 = 90.46%

TPR (Recall/Sensitivity):

TPR = TP / (TP + FN)
= 205 / (205 + 57)
= 205 / 262
= 0.7824 = 78.24%

TNR (Specificity):

TNR = TN / (TN + FP)
= 677 / (677 + 36)
= 677 / 713
= 0.9495 = 94.95%

PPV (Precision):

PPV = TP / (TP + FP)
= 205 / (205 + 36)
= 205 / 241
= 0.8506 = 85.06%

F1 Score:

F1 = 2 × (PPV × TPR) / (PPV + TPR)
= 2 × (0.8506 × 0.7824) / (0.8506 + 0.7824)
= 2 × 0.6656 / 1.633
= 0.8151 = 81.51%

Complete Results for All Classes:

Class	TP	FP	FN	TN	Accuracy	TPR	TNR	PPV	F1
Circle	205	36	57	677	90.46%	78.24%	94.95%	85.06%	81.51%
Idle	199	35	38	703	92.51%	83.97%	95.26%	85.04%	84.50%
Left-Right	223	4	60	688	93.44%	78.80%	99.42%	98.24%	87.40%

Up-Down	186	112	32	645	85.23%	85.32%	85.20%	62.42%	72.09%

11.3 Overall Accuracy

Total TP = 205 + 199 + 223 + 186 = 813
Total Samples = 262 + 237 + 283 + 218 = 1000

Overall Accuracy = Total TP / Total Samples
= 813 / 1000
= 0.813 = 81.3%

---

12. Embedded AI Deployment

12.1 Deployment Pipeline (From Exam Q4)

Step 1: Data Acquisition

→ Collect representative data
→ Label accurately
→ Balance classes
→ Split: Train (70%), Validation (15%), Test (15%)

Step 2: Model Training

→ Choose architecture (CNN, RNN, etc.)
→ Train on desktop/cloud
→ Optimize hyperparameters
→ Achieve target accuracy

Step 3: Model Optimization

→ Quantization (32-bit → 8-bit)
→ Pruning (remove unnecessary weights)
→ Knowledge distillation
→ Use Vela optimizer (for Ethos-U)

Step 4: Conversion

→ Convert to TensorFlow Lite (.tflite)
→ Optimize for embedded target
→ Verify accuracy maintained

Step 5: Deployment

→ Flash model to device
→ Integrate with application code
→ Test on real hardware
→ Monitor performance

Step 6: Handle Issues

a) Model Fails for Test Data:

Diagnosis:
- Overfitting to training data
- Insufficient diverse training samples
- Model too complex for task

Solutions:
1. Collect more diverse test data
2. Apply data augmentation
3. Use regularization (dropout, L2)
4. Simplify model architecture
5. Increase validation set

b) Low Accuracy on Smartphone + Nano Combined:

Diagnosis:
- Quantization errors
- Hardware incompatibility
- Different sensor characteristics

Solutions:
1. Re-quantize with representative data from both devices
2. Use quantization-aware training
3. Calibrate sensors
4. Create device-specific models
5. Use transfer learning

12.2 Performance Monitoring

Key Metrics:

1. Inference Time: <100ms for real-time
2. Power Consumption: <10mW for always-on
3. Memory Usage: <512KB RAM typical
4. Accuracy: >90% for production
5. False Positive Rate: <5%

Summary: Key Formulas

ADC & Sampling

Quantization Levels = 2^N
Resolution = $V_{ref} / 2^N$
Sampling Rate ≥ 2 × f_{max} (Nyquist)

Data Rate = Sampling Rate × Bits × Channels

Confusion Matrix

Accuracy = (TP + TN) / Total
TPR (Recall) = TP / (TP + FN)
TNR (Specificity) = TN / (TN + FP)
PPV (Precision) = TP / (TP + FP)

F1 Score = 2 × (PPV × TPR) / (PPV + TPR)

Signal Processing

FFT Resolution = Sampling Rate / FFT Size
Mel(f) = 2595 × log10(1 + f/700)

Timer Frequency = Clock / Prescaler
