Frage

1. Kontinuierlich vs. diskret:

Ein kontinuierliches Signal hat unendlich viele Werte im Zeitverlauf, ein diskretes Signal nur zu festen Zeitpunkten. Mikrocontroller arbeiten mit diskreten Werten, daher müssen kontinuierliche Signale z. B. über ADCs digitalisiert werden.

2. Nyquist-Shannon-Theorem:

Es besagt, dass ein Signal nur dann fehlerfrei rekonstruiert werden kann, wenn es mit mindestens der doppelten Frequenz seiner höchsten enthaltenen Frequenz abgetastet wird. Relevanz: Vermeidung von Aliasing beim Sampling.

3. Regelkreis:

Ein Regelkreis vergleicht eine Ist-Größe mit einer Soll-Größe und korrigiert Abweichungen über einen Regler. Er wird z. B. in Temperaturregelungen oder Motorsteuerungen verwendet.

4. Besonderheiten eingebetteter Software:

Sie muss ressourcensparend, echtzeitfähig und meist sehr zuverlässig sein. Im Gegensatz zur normalen Softwareentwicklung liegt hier der Fokus stärker auf Effizienz und Hardwarenähe.

5. Polling vs. Interrupt:

Polling fragt ständig nach einem Ereignis – einfach, aber ineffizient. Interrupts reagieren sofort bei Ereigniseintritt – effizienter, aber komplexer. Interrupts werden bei zeitkritischen Aufgaben bevorzugt.

6. Echtzeit-System & hart/weich:

Ein Echtzeitsystem reagiert innerhalb definierter Zeitgrenzen. Bei harter Echtzeit dürfen Fristen nie überschritten werden (z. B. Airbag), bei weicher sind Verzögerungen tolerierbar (z. B. Audio).

7. Echtzeit-Paradigmen:

Zeitgesteuert: Feste Zeitfenster, gut planbar, aber unflexibel.

Ereignisgesteuert: Reaktion auf Interrupts, flexibel, aber schwieriger zu testen.

8. Punkt-zu-Punkt vs. Bus:

Punkt-zu-Punkt ist störungssicher und echtzeitfähig, aber verkabelungsaufwendig. Bus spart Platz und Pins, ist aber störanfälliger. Eingebettete Systeme nutzen meist Bussysteme wie I^2C oder SPI.

9. Asynchron vs. synchroner Bus:

Synchron: Datenübertragung durch gemeinsamen Takt (z. B. SPI).

Asynchron: Kein gemeinsamer Takt, Start-/Stoppbits (z. B. UART). Synchron ist schneller, asynchron flexibler.

10. Fünf CPU-Phasen:

1. Fetch: Lade Befehl.
2. Decode: Analysiere Befehl.
3. Load: Hole Operanden.
4. Execute: Führe Operation aus.
5. Store: Schreibe Ergebnis zurück.

11. Cache-Prinzip:

Es nutzt Lokalität aus: häufig genutzte Daten liegen näher an der CPU. Unterstützung durch arrays statt Listen, kleine Schleifen, Daten zusammen speichern.

12. Assembler:

Maschinennaher Code in symbolischer Form, direkt übersetzbar in Maschinencode. Jede Prozessorarchitektur hat ihren eigenen Assembler.

13. Disassembler vs. Dekompilierer:

Disassembler wandelt Binärcode in Assembler, Dekompilierer in Hochsprache. Disassembler funktioniert bei jeder CPU gut, Dekompilierer besser bei Hochsprachen mit Metadaten.

14. Assembler-Berührungspunkte:

Startup-Code, Interrupt-Vektoren, Optimierungen, Hardwarezugriffe oder Debugging.

15. Eingebettetes System:

Ein spezialisiertes Computersystem, fest in ein technisches Produkt integriert (z. B. Waschmaschine, Auto). Nutzt Mikrocontroller oder kleine CPUs.

16. PID-Regler:

Besteht aus Proportional-, Integral- und Differentialanteil. Korrigiert Regelabweichungen durch Kombination aus aktueller, vergangener und erwarteter Abweichung – z. B. bei Motorregelung.

17. RISC vs. CISC:

RISC: Wenige, einfache Befehle, schneller.

CISC: Komplexe Befehle, geringer Programmcode. RISC für Effizienz, CISC bei Kompatibilität.

18. Signal:

Ein Signal ist eine zeitabhängige Veränderung einer physikalischen Größe, z. B. Spannung. Es transportiert Informationen zwischen Komponenten.

19. PWM-Signal:

Ein digitales Rechtecksignal mit variabler Pulsbreite. Durch Mittelwertbildung wirkt es wie ein analoges Signal – z. B. zur Motorsteuerung oder LED-Dimmung.

20. Quantisierungsrauschen:

Fehler, der durch Rundung bei der Digitalisierung entsteht. Tritt beim A/D-Wandeln auf und führt zu Verzerrungen im Signal.