PROJECT DISCLAIMER & TERMS OF USE

Source: M5Board / NAM3Forum / GitHub Community Project

1. **Community Standards:** This documentation is a community-driven effort. It is provided **free of charge** and is **not for sale or resale** under any circumstances. Contributions, corrections, and translations are encouraged—please push updates back to the community GitHub.
2. **As-Is Basis:** This XDF project is shared with the M5Board and NAM3Forum communities. It is provided as-is, and you use it entirely at your own risk. I accept no responsibility for any damage, loss, or issues resulting from the use or misuse of this file.
3. **Ghidra Integration:** Users are strongly encouraged to use the **Ghidra project folder** to cross-examine function logic and memory offsets. If you are unfamiliar with Ghidra, please use this project as an opportunity to educate yourself on its use for understanding the MS_S60 binary code. **Please contribute your Ghidra findings back to the community thread** so everyone can benefit from identified function addresses and logic flows.
4. **Technical Warning:** This document rephrases the BMW MS_S60 (S65/S85) Program Stand Description. Modifying ECU binary data (DME calibration) requires advanced knowledge of internal combustion physics. Always verify variable names (e.g., `K_...`, `L_...`) against the original German source, your Ghidra workspace, and the official XDF definitions before flashing.
5. **This information is provided for educational and off-road use only. Modifying emissions-related systems (e.g., catalyst heating, oxygen sensors) may be illegal in many areas and can impact the environment. Please consider the environmental effects and applicable laws before making any changes.**

   Any modifications are done at your own risk. The author assumes no responsibility for hardware damage, environmental impact, or legal consequences.

[Section 1] Module Overview and Strategy

The "Katalysatorheizen" (KH) function is a sub-module of the Exhaust Treatment system. Its primary purpose is to bring the catalytic converters to their operational "light-off" temperature as quickly as possible following a start.

Primary Heating Methods Controlled: The MS_S60 uses several interventions to increase exhaust gas enthalpy:

1. **Ignition Timing:** Significant retardation of the ignition angle (Wirkungsgradeingriff).
2. **VANOS:** Specific camshaft overlaps to increase exhaust temperature.
3. **Engine Speed:** A target idle increase (handled via the Idle Control module, but requested here).
4. **Secondary Air Injection (SLP):** Coordinating fresh air injection into the exhaust to react with rich exhaust gases.

Master Status Bit: The logic determines the state of the variable **`kh_st`** (Catalyst Heating Status). When `kh_st` is active, it triggers the offsets for ignition and VANOS.

[Section 2] Logic Flow: Temperature Selection (Temperaturauswahl)

Before activation, the ECU must determine if heating is even necessary based on the current thermal state.

Logic Actions (Block Diagram Description):

- **Inputs:** The system monitors `tmot` (Coolant Temp), `tans` (Intake Air Temp), and Ambient Temperature.
- **Case Selection (Fallunterscheidung):**
  - **Case A (Standard):** If the CAN bus signal for ambient temperature (`b_can_st_atemp_ok`) is valid, it uses that value to index the heating maps.
  - **Case B (Fail-safe):** If the CAN signal is missing, the system defaults to the constant **`K_KH_TKAT_CFG`**.
- **Comparison:** The selected temperature is compared against **`K_KH_TKAT_ENDE`** (Heating End Threshold).
- **Hysteresis:** To prevent the system from "flickering" on and off, a hysteresis value (**`K_KH_TKAT_HYST`**) is applied. If the temperature is above the threshold plus hysteresis, heating is suppressed.

[Section 3] Logic Flow: Release Conditions (Freigabe)

Even if the engine is cold, heating will be blocked if certain "Inhibitor" conditions are met to ensure engine safety and drivability.

Logic Actions (Block Diagram Description):

1. **RPM Check:** Current engine speed (`n`) is compared against **`K_KH_N_MAX`**. If `n > K_KH_N_MAX`, the bit `b_kh_aus_n` is set, and heating is aborted.
2. **Battery Voltage:** If system voltage is too low, heating is disabled to reduce alternator load.
3. **Load Check:** If the driver requests high torque (Relative Load `rl` > **`K_KH_RF_MAX`**), the heating is immediately terminated to provide full power.
4. **Time Counter:** The system tracks `kh_time_on`. If the duration exceeds **`K_KH_TIME_DCY_MAX`**, the cycle ends regardless of temperature.

Ghidra Note: In the code, these checks appear as a series of conditional branches (if/else) that must all return "True" to maintain the `kh_st` active bit. You can find this sub-routine by searching for the memory address associated with **`K_KH_N_MAX`**.

[Section 4] Dynamic Detection (Dynamikerkennung)

During the catalyst heating phase, the engine's torque delivery is less efficient due to retarded ignition. To prevent drivability issues during sudden movements, the ECU monitors "Dynamics" (rapid changes in pedal position or load).

Logic Actions (Block Diagram Description):

- **Input Monitoring:** The system monitors the derivative (rate of change) of the throttle or load signal.
- **Hold Function:** If a sudden load change is detected, the bit **`b_kh_dyn`** is set.
- **The "Timer" Gate:** The system uses **`K_KH_DYN_T_HOLD`** (Dynamic Hold Time). This timer ensures that even after the driver stops moving the pedal, the ignition timing doesn't snap back instantly; it stays in a "hold" state to stabilize combustion.
- **XDF Reference:** Adjusting `K_KH_DYN_T_HOLD` affects how sensitive the car is to "canceling" the heating mode when you blip the throttle.

[Section 5] VANOS Target Intervention (Sollwertbeeinflussung VANOS)

To maximize heat flow to the cats, the ECU alters the camshaft timing specifically for the heating phase.

Logic Actions (Block Diagram Description):

- **Base vs. Offset:** The ECU takes the standard VANOS maps and adds/subtracts an offset specifically for heating.
- **Intake (E):** The target position is adjusted by **K_VAN_KH_ABR_E**.
- **Exhaust (A):** The target position is adjusted by **K_VAN_KH_ABR_A**.
- **Bank Smoothing:** In the S85/S65, this is calculated using the **Index [j]**. The ECU ensures Bank 1 and Bank 2 are synchronized, but checks individual bank errors (`b_van_nr_err[j]`). If one bank has a VANOS fault, the heating intervention for that bank's camshafts may be disabled to protect the engine.
- **Ghidra Tip:** Look for the function that sums the `van_soll` (base target) with the `kh_van_offset`.

[Section 6] Efficiency Intervention: Ignition Retard (Wirkungsgradeingriff)

This is the "core" of the heating process. By retarding the ignition, more combustion energy is sent out of the exhaust valve as heat rather than being converted into mechanical work at the piston.

Logic Actions (Block Diagram Description):

- **Torque Reserve:** Because the ignition is retarded, the engine produces less torque. To prevent stalling, the Idle Control system must open the idle actuators or throttle butterflies slightly more. This "extra air" is calculated as a torque reserve.
- **Calculation:** The efficiency factor **etakh** is calculated. A value of 1.0 is full efficiency; during heating, this might drop significantly (e.g., 0.6 or 0.7).
- **Ramping:** When the heating cycle ends, the ignition timing does not "jump" back to the fuel-efficient setting. It follows a ramp defined by **K_KH_ETAKH_RAMPE** to ensure the driver feels a smooth increase in power rather than a sudden surge.

[Section 7] Status Output (1.6 Zustandsausgabe)

Once all the internal logic has been processed, the system needs to "broadcast" the result to the rest of the engine management system. This ensures that modules like the Idle Control or the SMG/DCT Transmission know that the engine is currently in a specialized thermal state.

Logic Actions (Block Diagram Description):

- **The Master Flag:** The bit `b_kh_st` (Catalyst Heating Status) is the final output of the logic gates from the previous pages.
- **Inter-Module Communication:**
  - **To Injection:** Signals that the air-fuel ratio may need to be adjusted (Leaning or Enriching depending on Secondary Air status).
  - **To Idle Control:** Passes the torque reserve requirement (`etakh`) so the throttle plates can compensate for the retarded timing.
  - **To CAN Bus:** Sends the heating status to the cluster or diagnostic tools so a tuner or technician can see if the system is active.
- **Error Handling:** If a sensor required for the calculation (like `tmot`) fails, the status output defaults to "OFF" or a safe "Emergency Heating" state defined by `K_KH_ERR_MODE`.

[Section 8] Secondary Air Injection Coordination (Sekundärlufteinblasung - Intro)

While "Catalyst Heating" is a software strategy, **Secondary Air Injection (SLP)** is the physical hardware (pump and valves) used to accelerate the chemical reaction in the exhaust.

Logic Actions (Block Diagram Description):

- **Synchronization:** The KH module sends a "Request" to the SLP module. The SLP will only start if the KH module confirms that the ignition timing is retarded enough to provide unburned fuel for the secondary air to react with.
- **Variable `b_kh_slp_req`:** This is the trigger bit.
- **The "Oxygen Balance":** The system calculates how much extra air is being pumped in so it can prevent the Lambda sensors from thinking the engine is running lean and accidentally over-fueling (this is where the **Index [j]** becomes critical, as the pump may output slightly different flows to Bank 1 and Bank 2).

[Section 9] Component Protection & Exit Strategies

The ECU constantly looks for reasons to stop heating to prevent damage to the exhaust valves or the "washcoat" of the catalytic converters.

Logic Actions (Block Diagram Description):

- **EGT Modeling:** Since there is no physical sensor for Exhaust Gas Temperature (EGT) on many versions, the ECU uses a "Model" based on RPM, Load, and Timing.
- **The Safety Gate:** If the modeled temperature exceeds `K_KH_EGT_MAX`, the bit `b_kh_prot_active` is set.
- **Action:** This overrides all other logic and immediately ramps the ignition timing back to "Normal" (Optimal Efficiency) to cool the exhaust valves down.
- **Ghidra Note:** This logic is often found near the "Safety Monitoring" or "Component Protection" (Bauteilschutz) code blocks. If you see the code jumping to a routine that forces `etakh` to 1.0, you have found the Protection Exit.

[Section 10] Lambda Regulation Overview (Stetiger Lambdaregler)

Moving from the heating phase into standard operation, the **Lambda Controller** takes over to manage the air-fuel ratio. In the MS_S60, this is a "continuous" (stetiger) regulation system using wideband sensors.

Logic Strategy: The controller's job is to ensure the actual air-fuel ratio matches the target (`lam_soll`) by adjusting the injection mass. Because the S65 and S85 are high-performance engines, the regulation is extremely fast and bank-specific.

- **Bank Index [j]:** The ECU runs two independent control loops. All variables followed by `[j]` (e.g., `fr[j]`) are calculated separately for Bank 1 and Bank 2.
- **Target vs. Actual:** The system compares the target Lambda with the measured value from the LSU (Lambda Sensor Universal).

[Section 11] Sensor Signal Processing (Sondensignalauswertung)

Before the ECU can use the oxygen sensor data, it must validate the signal.

Logic Actions (Block Diagram Description):

1. **Read Raw Voltage:** The ECU reads the pump current/voltage from the wideband sensor.
2. **Characteristic Curve (XDF: `KL_LSU_...`):** The raw signal is passed through a linearization map to convert voltage into a Lambda value.
3. **Plausibility Check:**
    - If the sensor is too cold, the bit `b_ls_ber[j]` (Sensor Ready) is set to False.
    - If the signal is stuck at a specific voltage for too long, a "Stuck" diagnostic is triggered.
4. **Heating Control:** The ECU manages the sensor's internal heater to maintain a constant ceramic temperature (approx. 750-800°C). If the heater fails, the Lambda control for that bank drops into "Open Loop" mode.

[Section 12] Controller Release & Mode Selection (Freigabe Lambdaregelung)

"Closed Loop" operation is not always active. The ECU uses a complex set of "AND" gates to decide when to trust the sensors.

Logic Actions (Block Diagram Description):

- **Conditions for Activation:**
    - `tmot` > `K_T_LAM_MIN`: The engine must be above a minimum temperature.
    - `b_ls_ber[j]` = **True**: The sensor must be heated and ready.
    - `kh_st` = **False**: Usually, standard Lambda regulation is suspended or modified during the heavy ignition retard of the Catalyst Heating phase.
- **The Status Bit `b_lr[j]`:** This is the master flag for "Lambda Regulation Active."
    - **0 (Open Loop):** The ECU uses pre-defined maps (calculated from air mass and RPM).
    - **1 (Closed Loop):** The ECU uses the sensor feedback to trim the fuel in real-time.
- **Ghidra Note:** You can trace the `b_lr` bit in the binary to find the logic that switches between "Map-based" fueling and "Feedback-based" fueling.

[Section 13] Main Control Parameter: `fr[j]` (Regelfaktor)

The most important output of this module for tuners is the `fr[j]` (Lambda Control Factor).

Technical Definition:

- `fr = 1.00`: No correction (The engine is hitting its target exactly).
- `fr > 1.00`: The ECU is adding fuel (The sensor detected a lean condition).
- `fr < 1.00`: The ECU is removing fuel (The sensor detected a rich condition).

Logic Actions:

- **P-Part and I-Part:** The controller is a PI (Proportional-Integral) controller.
    - **Proportional:** Reacts immediately to large deviations.
    - **Integral:** Slowly "walks" the fuel trim to zero out persistent errors.
- **Limits:** The correction is capped by `K_FR_MAX` and `K_FR_MIN` (typically +/- 25%). If the ECU hits these limits and the target is still not met, it will trigger a "System Lean/Rich" check engine light.

[Section 14] Individual Cylinder Lambda Correction (Einzelzylinderregelung)

Beyond bank-specific control, the MS_S60 employs a sophisticated algorithm to balance fueling for each individual cylinder. This is critical for the high-revving S65/S85 to ensure smooth idle and prevent localized lean conditions that could damage a single piston.

Logic Actions (Block Diagram Description):

- **Segment Time Analysis:** The ECU measures the "Segment Time" (the time it takes the crankshaft to rotate through a specific degree range) after each cylinder fires.
- **Torque Contribution:** If one cylinder is "faster" (shorter segment time), the ECU identifies it as producing more torque (likely richer). If it is "slower," it is likely leaner.
- **The Correction Factor `fac_lam_cyl[i]`:** * **Index [i]:** Refers to the specific cylinder (1–10).
    - **Action:** The system applies a micro-adjustment to the injection pulse width for that specific cylinder to equalize its torque contribution with the others.
- **Release Gate:** This function is typically only active at lower RPMs and idle. It is disabled at high RPM where crankshaft torsion makes segment time analysis unreliable.

[Section 15] Lambda Adaptation / Learning Values (Lambda-Adaption)

To compensate for long-term changes (like vacuum leaks, aging injectors, or fuel quality), the ECU "learns" a permanent correction. This is what prevents the `fr[j]` (Real-time factor) from having to work too hard.

Technical Variables (XDF Reference):

1. **Additive Adaptation (`laa_offs[j]`):** Corrects errors that are constant, regardless of load (e.g., a small vacuum leak at idle). Measured in milliseconds of injection time.
2. **Multiplicative Adaptation (`laa_f[j]`):** Corrects errors that scale with fuel flow (e.g., fuel pump pressure deviation or injector scaling). Measured as a percentage factor.

Logic Actions:

- **Learning Conditions:** The ECU only updates these values when the engine is in a stable "Steady State" (constant RPM and load).
- **Storage:** These values are stored in the EEPROM/Non-volatile memory. If you disconnect the battery or flash a new tune, these are often reset to 0.0 or 1.0.
- **Ghidra Tip:** The adaptation logic is a "Slow Path" in the code. It uses a long-term integrator that only increments when the `b_laa_ready` flag is set.

[Section 16] Lambda Change Limitation (Begrenzung Lambdaänderung)

To ensure drivability and prevent the driver from feeling "surges" during transitions (like moving from Catalyst Heating to Normal Mode), the ECU limits how fast the Lambda target can change.

Logic Actions (Block Diagram Description):

- **Input:** The raw target Lambda (`lam_soll_raw`).
- **The Smoother:** The system applies a gradient limit defined by **`K_LAM_GRAD_MAX`**.
- **Result:** The final **`lam_soll`** follows a smooth ramp. This prevents the torque from jumping abruptly, which could cause the car to jerk or the dual-clutch transmission (DCT) to behave unexpectedly during a shift.

[Section 17] Sensor Protection & Dew Point (Taupunkterkennung)

Wideband sensors are fragile. If they are heated while liquid water (condensation) is present in the exhaust, the ceramic element will crack (thermal shock).

Logic Actions:

- **Dew Point Model:** The ECU calculates the "Dew Point" temperature of the exhaust based on `tmot` and the time since the engine started.
- **Heating Delay:** The variable `K_LSU_HEIZ_DELAY` keeps the sensor heaters off until the exhaust is guaranteed to be dry.
- **Bit Status `b_ls_he_rel[j]`:** This bit must be high before the sensor heater is energized.

[Section 18] Secondary Air Injection Overview (Sekundärlufteinblasung)

The **Secondary Air Injection (SLP)** system is the hardware counterpart to the Catalyst Heating software strategy. By injecting fresh air into the exhaust tract upstream of the catalytic converters, the ECU promotes the post-oxidation of unburned hydrocarbons (HC) and carbon monoxide (CO). This exothermic reaction generates the thermal energy required to bring the catalysts to "light-off" temperature rapidly.

System Components:

- **Electric Air Pump (SLP):** Supplies the required fresh air mass flow.
- **Secondary Air Valve:** A non-return/switch valve that prevents exhaust backflow into the pump and opens the path for air injection.
- **Mini-Hot-Film Air Mass Meter (mHF):** Measures the actual secondary air mass flow to allow the ECU to monitor system performance and adjust fueling.

[Section 19] Control Logic & Activation (Ansteuerung und Freigabe)

The SLP system is only activated during the specific "warm-up" window determined by the Catalyst Heating module.

Logic Actions (Block Diagram Description):

1. **Request Input:** The system waits for the bit `b_kh_slp_req` (Secondary Air Request from Catalyst Heating) to be set to True.
2. **Environmental Permissives:**
    - **Voltage Check:** Battery voltage must be above `K_SLP_U_MIN` to ensure the high-draw electric pump doesn't brown out the electronics.

    - **Ambient Pressure:** At high altitudes, pump efficiency drops. The constant `K_SLP_PUMG_MIN` defines the minimum atmospheric pressure required for operation.
3. **The Master Bit `b_slp`:** If all conditions are met, the pump is energized.
4. **Ghidra Note:** The relay activation is usually a direct I/O trigger. Searching for the address of the bit `b_slp` in Ghidra will lead you to the physical pin assignment on the ECU connector.

[Section 20] Mass Flow Calculation & Lambda Compensation (Luftmassenberechnung)

When the air pump is running, the exhaust becomes artificially "lean" due to the added oxygen. The ECU must compensate for this to maintain the desired combustion mixture.

Logic Actions (Block Diagram Description):

- **Modeled Air Mass:** The ECU calculates the secondary air mass based on the pump's characteristic curve (`KL_SLP_ML`) and current system voltage.
- **The Compensation Factor:** This calculated mass is converted into a "Lambda Offset."
- **Action:** The ECU adjusts the internal Lambda calculation to account for this extra oxygen. This allows the engine to run physically rich (providing fuel for the exhaust reaction) while the Lambda controller maintains stability.
- **XDF Reference:** If the pump is physically removed without software adjustment, the fueling will be incorrect during the warm-up phase. You must address `KL_SLP_ML` or the `b_slp` trigger.

[Section 21] Secondary Air System Diagnosis (Diagnose)

The ECU must verify the system is functioning correctly by monitoring the effect of the air injection on the exhaust gas composition.

Logic Actions (Block Diagram Description):

- **Monitoring Period:** The diagnosis is active when the bit `b_slp` is set and the engine is in a stable state.
- **Sensor Feedback:** The system monitors the Lambda values (`lam[j]`) for Bank 1 and Bank 2.
- **Logic Gate:** If air is flowing, the Lambda value must shift in the lean direction. The system compares this shift against the threshold `K_SLS_L_DIFF_MIN`.
- **Diagnostic State:** The process is tracked via the state machine `sls_obd_diag_st`.

- **Fault Flag:** If the expected lean shift is not detected after the stabilization time `K_SLS_T_STAB`, the system sets the diagnostic error bit `b_ed_sl_system_fehler`.

[Section 22] Fuel Injection Logic (Einspritzung)

This module calculates the effective injection time (te) for each cylinder based on the required fuel mass to achieve the target Lambda.

Logic Actions (Block Diagram Description):

1. **Input:** The base fuel mass is derived from the relative filling (rl), which represents the air mass in the cylinder.
2. **Correction Factors:** The base time is multiplied by a chain of factors:
    - `fac_lam_soll`: The factor required to reach the target Lambda.
    - `fr[j]`: The real-time output from the Lambda controller (Bank-specific).
    - `laa_f[j]`: The multiplicative adaptation (learning) value (Bank-specific).
3. **Physical Limits:** The calculated pulse width is compared against `K_TI_MIN`. If the result is lower than this constant, the ECU forces the value to `K_TI_MIN` to ensure the injector can actually fire.
4. **Voltage Compensation:** Injector latency (dead time) is added. The ECU uses the characteristic curve `KL_TE_UB` to find the time offset based on battery voltage (ub).

[Section 23] Injection Timing & End of Injection (Spritzbeginn)

This page details the timing of the fuel spray relative to the crankshaft position to optimize mixture preparation.

Logic Actions (Block Diagram Description):

- **Target Selection:** The system calculates either the `asoe` (Angle of Start of Injection) or the `aeoe` (Angle of End of Injection).
- **Map Lookup:** The primary calculation uses the map `KF_ASOE`, indexed by Engine Speed (n) and Load (rl).
- **Phase Correction:** During the engine start phase, an additive offset `K_ASOE_START` is applied. This shifts the timing to ensure fuel is sprayed onto the back of the intake valve while it is still closed, utilizing valve heat for better evaporation.
- **Ghidra Note:** The `asoe` calculation is handled in a high-priority task. Finding the map `KF_ASOE` in the data segment will allow you to see the factory-defined injection "window" across the RPM range.

[Section 24] Individual Cylinder Torque/Fuel Balancing (Laufunruheregelung)

The MS_S60 monitors the "roughness" of the engine to detect if specific cylinders are underperforming or over-fueling.

Logic Actions (Block Diagram Description):

1. **Segment Time Measurement:** The ECU measures the time taken for the crankshaft to rotate between specific tooth markers for each cylinder.
2. **Comparison:** The system compares the segment time of cylinder i against the average segment time of all cylinders.
3. **Correction:** If cylinder i is slower than the average, the bit `b_lur_active` allows the system to apply a correction factor to that specific cylinder's injection time.
4. **Limits:** The maximum correction is capped by `K_LUR_MAX`.
5. **Deactivation:** The function is disabled if the engine speed exceeds `K_LUR_N_MAX` or if a gear shift is detected (`b_schalt`), as these conditions create "artificial" roughness that would confuse the logic.

[Section 25] Throttle Valve and Idle Control (Drosselklappen und Leerlaufsteller)

The MS_S60 manages engine airflow through two primary actuator systems: the **Electronic Throttle Butterflies (DK)** and the **Idle Actuators (LLS)**. On the S65/S85, these must be perfectly synchronized across both banks.

Logic Actions (Block Diagram Description):

1. **Setpoint Calculation:** The driver's torque request (from the accelerator pedal) is converted into a target air mass.
2. **Actuator Split:** * **At Idle:** The logic primarily uses the **Idle Actuators (LLS)** to manage airflow for stability.
    - **Above Idle:** The **Throttle Butterflies (DK)** take over as the primary load control.
3. **Bank Synchronization (Index [j]):** The ECU calculates a target position for Bank 1 and Bank 2 simultaneously. The variables `dk_soll[j]` and `lls_soll[j]` are the final outputs.
4. **Monitoring:** The actual positions (`dk_ist[j]`) are monitored via dual-potentiometers. If a deviation between `soll` (target) and `ist` (actual) exceeds `K_DK_MAX_DIFF`, the system triggers a safety shutdown of the power stages.

[Section 26] Idle Speed Control (Leerlaufregelung)

The idle speed control logic ensures the engine maintains a stable RPM regardless of load changes (e.g., A/C compressor activation or power steering load).

Logic Actions (Block Diagram Description):

- **Target RPM:** The base idle target is defined by the map `KF_N_SOLL_LL`, indexed by coolant temperature (`tmot`).
- **Load Compensation:** If the Catalyst Heating (`kh_st`) is active, the idle target is increased by the offset `K_N_LL_KH`.
- **Controller Type:** This is a PID controller.
    - **Proportional/Integral:** Adjusts the **LLS** actuators for long-term stability.
    - **Differential:** Uses the **Ignition Angle** (fast path) to catch sudden RPM drops before the air actuators can react.
- **XDF Reference:** To change the target idle speed of the car, you must modify `KF_N_SOLL_LL`.

[Section 27] Throttle Valve Emergency Logic (Notlauf Drosselklappe)

Because the S65/S85 uses a "Linkless" electronic throttle, the ECU contains extensive safety code to prevent unintended acceleration.

Logic Actions (Block Diagram Description):

1. **Plausibility Check:** The ECU compares the air mass measured by the **HFM** (Mass Air Flow sensor) against the calculated air mass based on the **DK** position.
2. **Error Detection:** If the two values do not match (e.g., a throttle is stuck open but the pedal is released), the bit `b_dk_fehler` is set.
3. **Limp Home Mode:**
    - The ECU cuts power to the throttle motors.
    - High-tension springs pull the throttles to a "Default" slightly open position (`K_DK_NOT_POS`).
    - The engine speed is then limited by cutting fuel to specific cylinders (Fuel Cut) to prevent the RPM from climbing.
- **Ghidra Note:** The safety monitoring code is often mirrored in a second processor (Monitoring Processor) within the DME. Disabling these checks in the main code without addressing the monitor will lead to an immediate "Engine Fail-safe" mode.

[Section 28] Ignition Control Overview (Zündwinkelberechnung)

The ignition module calculates the final ignition angle (`zwout`) for each cylinder. This represents the actual timing of the spark in degrees Crankshaft (°KW) relative to Top Dead Center (TDC).

Technical Breakdown & Calculation: The final ignition angle is determined by a summation logic that starts with a base value and layers multiple corrections (offsets).

The Calculation Formula:

zwout=KF_ZW_GRUND(n,rl)+dzw_tmot+dzw_kh+dzw_kr+dzw_ll

Variables & Components:

- `KF_ZW_GRUND(n, rl)`: The primary 16x16 map. It defines the ignition timing based on Engine Speed (`n`) and Relative Filling (`rl`).
- `dzw_tmot`: A temperature-dependent offset. It advances or retards the timing based on coolant temperature to stabilize combustion during the warm-up phase.
- `dzw_kh`: The **Catalyst Heating** offset. When the bit `b_kh` is active, this value significantly retards the timing to shift heat into the exhaust.
- `dzw_kr`: The **Knock Control** offset (derived from `dwk[i]`). This is a negative value (retard) applied when the acoustic or ion-current sensors detect pre-ignition.
- `dzw_ll`: The **Idle Speed Control** intervention. This is a "fast-path" correction used to stabilize RPM by quickly advancing or retarding timing before the air actuators (DK/LLS) can react.

Safety Limiting (Begrenzung): After the summation, the logic passes `zwout` through a window comparator:

- `K_ZW_MIN`: The "Late Limit." Prevents timing from retarding so far that it causes a misfire or exhaust valve damage.
- `K_ZW_MAX`: The "Early Limit." The maximum allowable advance to prevent structural engine damage, regardless of other inputs.

[Section 32] VANOS Control (Variable Nockenwellensteuerung)

The VANOS system manages the phase shift of the intake and exhaust camshafts to optimize cylinder filling and exhaust gas scavenging.

Technical Breakdown & Calculation: The ECU calculates a pulse-width modulated (PWM) signal to drive the hydraulic solenoids, based on the deviation between target and actual positions.

The Calculation Formula for Control Deviation:

$$\Delta van[j] = van\_soll[j] - van\_ist[j]$$

Variables & Components:

- `van_soll[j]`: The target position calculated from `KF_VAN_SOLL_E` (Intake) and `KF_VAN_SOLL_A` (Exhaust).
- `van_ist[j]`: The actual camshaft position measured by the Hall-effect sensors on Bank `[j]`.
- `K_VAN_I_HOLD`: The stationary current required to maintain a cam position against hydraulic drift.

Logic Actions:

1. **Setpoint Adjustment:** If Catalyst Heating is active, the target is modified: $van\_soll\_active = van\_soll + K\_VAN\_KH\_ABR$
2. **Controller Output:** The PID controller calculates the current (I) required to close the gap $\Delta van[j]$.
3. **Diagnosis:** If $|\Delta van[j]| > K\_VAN\_TOL$ for a duration longer than `K_VAN_T_MAX`, the bit `b_van_nr_err[j]` is set, and the system enters a fail-safe state.

[Section 33] Fuel Tank Ventilation (9.1 Tankentlüftung - TE)

This module manages the introduction of fuel vapors from the carbon canister into the intake manifold while maintaining Lambda stability.

Technical Breakdown & Calculation: The ECU must calculate the "purge mass" to adjust the fuel injection pulse width, ensuring the extra fuel vapor doesn't cause a rich condition.

The Calculation for Injection Correction:

$$teeff = tebase \times (1 - fac\_te)$$

Variables & Components:

- `fac_te`: The "Tank Ventilation Factor." This represents the percentage of total fuel mass currently being supplied by the purge vapors.

- **b_te_active**: The master bit that enables the TE solenoid valve (**TEV**).
- **KL_TEV_KR**: A characteristic curve that defines the flow rate of the purge valve based on the duty cycle and intake manifold vacuum.

Logic Actions:

1. **Regeneration:** The ECU periodically opens the TEV to "clean" the carbon canister.
2. **Lambda Adaptation:** During purge, the ECU monitors the Lambda controller factor (**fr[j]**). If fr drops (indicating a rich condition), the ECU calculates the concentration of the vapor and updates **fac_te**.
3. **Blocking:** Purge is blocked if the engine is in a critical state (e.g., full load or during Catalyst Heating) to prevent enrichment errors.

[Section 34] Engine Oil System & Scavenge Pump Control (Motoröl- und Saugpumpensteuerung)

The MS_S60 manages a complex oiling system required for high-lateral-G maneuvers, specifically the S85 V10's multi-pump setup. This includes the main pressure pump and the electric scavenge pumps.

Technical Breakdown & Calculation: The ECU regulates the delivery rate of the oil pumps to maintain target pressure while minimizing parasitic drag on the engine.

The Calculation for Target Oil Pressure:

p_oel_soll=KF_POEL_SOLL(n,tmot)+dpoel_stat

Variables & Components:

- **KF_POEL_SOLL**: A 16x16 map defining required oil pressure based on Engine Speed (**n**) and Coolant Temperature (**tmot**).
- **p_oel_ist**: The actual oil pressure measured by the pressure sensor.
- **b_p_oel_mini**: A critical safety bit. If p_oel_ist falls below the minimum threshold defined in **K_POEL_MIN**, the engine enters an emergency shutdown or power reduction mode.
- **b_saugp_en**: Enable bit for the electric oil scavenge pumps. These are triggered based on lateral acceleration (**ay**) signals received via the CAN bus from the DSC module.

Logic Actions:

1. **Dynamic Regulation:** The ECU adjusts the volume flow of the oil pump (via a solenoid valve on the pump) to match p_oel_soll.

2. **Scavenge Trigger:** During high-speed cornering, the ECU activates the lateral scavenge pumps to prevent oil from pooling in the cylinder heads, ensuring the main sump always has a supply.
3. **Level Monitoring:** The electronic oil level sensor provides a signal (`oel_stand`) that is filtered over time to prevent false warnings during spirited driving.

[Section 35] Cooling System & Map Thermostat (Kühlsystem / Kennfeldthermostat)

The MS_S60 utilizes a "Map-Controlled" thermostat that allows the ECU to artificially increase or decrease the engine's operating temperature for efficiency or performance.

Technical Breakdown & Calculation: The ECU controls a heating element inside the thermostat to "trick" it into opening earlier than its mechanical rating.

The Calculation for Thermostat Duty Cycle:

pwm_them=KL_THEM_SOLL(n,rl)+dthem_korr

Variables & Components:

- `KL_THEM_SOLL`: A characteristic curve mapping the target temperature to the heating element's PWM duty cycle.
- `tmot`: The actual coolant temperature.
- `K_TMOT_MAX`: The maximum allowable temperature before the electric fan is forced to 100% duty cycle.

Logic Actions:

1. **Economy Mode:** Under light load, the ECU allows `tmot` to rise (approx. 100°C–105°C) to reduce internal friction and improve fuel economy.
2. **Performance Mode:** When high load (`rl`) or high RPM (`n`) is detected, the ECU energizes the thermostat heater to drop `tmot` (approx. 75°C–85°C), increasing the knock margin and protecting the engine.
3. **Fan Control:** The electric cooling fan is driven by a PWM signal determined by the radiator outlet temperature sensor and A/C system pressure.

[Section 36] Alternator and Energy Management (Generatorregelung / BSD)

The ECU communicates with the alternator via the **BSD (Bit-Serial Data)** interface to manage electrical load and battery charging.

Technical Breakdown & Calculation: The target charging voltage is dynamically adjusted based on the battery's state of charge and temperature.

The Calculation for Target Voltage:

$$u\_gen\_soll = KL\_U\_GEN\_TEMP(tbat) + du\_gen\_load$$

Variables & Components:

- `KL_U_GEN_TEMP`: A curve defining the charging voltage (typically 13.5V–15.0V) based on the battery temperature (`tbat`).
- `tbat`: Battery temperature, usually provided by the IBS (Intelligent Battery Sensor).
- `b_gen_err`: A fault bit set if the alternator fails to respond to BSD commands or cannot reach the target voltage.

Logic Actions:

1. **Load Shedding:** If the engine is under maximum load, the ECU can temporarily reduce the alternator's output to maximize the torque available to the wheels.
2. **Overrun Recovery:** During deceleration (coasting), the ECU increases the generator voltage to "capture" energy and charge the battery without using fuel.

[Section 37] Diagnostics and OBD-II Monitoring (Fehlerspeicher / OBD)

This module manages the detection, debouncing, and permanent storage of faults. It distinguishes between a "sporadic" glitch and a "hard" component failure.

Technical Breakdown & Logic: The calculation relies on an **increment/decrement counter** system. Instead of triggering a fault immediately, the ECU starts a timer-based counter. If the error condition is true, the counter increases; if the error disappears, the counter slowly decreases. Only when this counter hits a pre-defined maximum threshold (`K_ERR_LIMIT_HI`) is the fault considered "Confirmed."

Ghidra Reference: Look for the **Diagnostic State Machine** in the code. You can find this by searching for references to the `DTC` **(Diagnostic Trouble Code)** memory area. The logic usually

involves a large `switch-case` statement where each case corresponds to a specific sensor's debouncing counter. The function that updates the fault memory is typically called when the bit **`b_em_mil`** (MIL Request) transitions from 0 to 1.

[Section 38] Misfire Detection (Aussetzererkennung)

This is a safety-critical module that protects the catalytic converters. On the S65/S85, it cross-references mechanical vibration with electrical ion current.

Technical Breakdown & Logic: The calculation is based on **Crankshaft Roughness**. The ECU measures the time it takes for the crankshaft to rotate a specific "segment" (the degrees between cylinder firings).

1. The ECU compares the current segment time to the previous one.
2. If a cylinder misfires, it fails to push the piston down, causing the crankshaft to decelerate.
3. This creates a "long" segment time.
4. The calculation subtracts the expected time from the actual time and cubes the result to amplify the difference, creating a **Roughness Value**. If this value exceeds the threshold in the map **`K_LU_LIMIT_N`**, a misfire is logged.

Ghidra Reference: The misfire logic is tied to the **Crankshaft Position Sensor (KWG)** interrupt routine. Search for code that accesses the segment time variables (usually named **`T_SEG`** or similar). The logic that sets the injection cut-off bit **`b_st_enz[i]`** is the "Action" part of this function. You can find it by tracing where the ECU executes a "Stop" command to the injector output drivers.

[Section 39] Cruise Control and Speed Limiter (Geschwindigkeitsregelung / FGR)

This module acts as a "Secondary Driver," requesting torque to maintain a constant vehicle speed.

Technical Breakdown & Logic: The calculation uses **Proportional-Integral-Derivative (PID)** logic.

- It looks at the "Error" (the difference between your set speed and actual speed).
- It calculates how much torque is needed to close that gap.
- If you are going uphill, the "Integral" part of the calculation grows larger over time, adding more and more throttle until the speed stabilizes.
- The final result is a **Torque Request** sent to the Torque Coordinator, which then decides whether to open the throttles or adjust ignition.

Ghidra Reference: Search for the variable `v_soll` (Target Speed). The FGR logic is a separate task that runs at a slower rate (often every 10ms or 20ms). In the binary, this section contains the logic for the `K_VMAX` (Top Speed Limiter). To find the VMAX limiter, look for a comparison between the current vehicle speed and a constant (usually 250 hex or similar), followed by a conditional branch that forces a torque reduction.

[Section 40] Coding and Variants (Codierung / Varianten)

The MS_S60 is a "Global" ECU. It contains the logic for all regions (US, EU, JP) and all vehicle types (M3, M5, M6) in a single binary.

Technical Breakdown & Logic: The ECU uses a **Coding Word** calculation. During the boot-up phase, the ECU reads a specific area of the EEPROM.

- It uses a "Masking" logic (Bitwise AND/OR) to check which bits are set.
- For example, if Bit 3 is "1," the ECU enables the **US-Specific** Secondary Air diagnostics.
- If Bit 5 is "1," it enables the **V10 (S85)** cylinder constants instead of the **V8 (S65)** constants.
- This calculation effectively "turns off" large blocks of code so they never execute.

Ghidra Reference: This is the most important area for "Option Deleting" (like removing SAP or Cat sensors). Look for the initialization routine at the very beginning of the code. You will see a series of checks against a variable often called `C_VARIANT` or `Codierwort`. By changing the results of these checks in Ghidra, you can force the ECU to skip the diagnostic routines for specific hardware components.

[Section 41] Data Management and Checksums (Datenhaltung / Prüfsummen)

To ensure the integrity of the 3,000+ pages of logic and thousands of maps, the MS_S60 uses a tiered checksum system.

Technical Breakdown & Logic: The calculation is a **Cyclic Redundancy Check (CRC)**.

1. The ECU does not just sum the bytes; it uses a polynomial math logic to create a unique "Fingerprint" for different blocks of memory (Program Code, Calibration Data, and Variable RAM).
2. During the "Pre-Drive" check, the ECU re-calculates the fingerprint of the Calibration block and compares it to the stored value.
3. If they do not match (e.g., if a tuner changed a map but didn't update the checksum), the ECU sets the bit `b_ck_err` and prevents the engine from starting to avoid unpredictable behavior.

Ghidra Reference: In Ghidra, you will find the checksum routine near the end of the initialization sequence. It usually involves a loop that iterates through a memory range (defined

by a start and end address pointer) and performs a series of bit-shifts and XOR operations. The result is then compared to a value stored at a specific "Checksum Anchor" address.

[Section 42] Torque Coordinator (Momentenstruktur)

This is the "Brain" that sits above all other modules. It resolves conflicts between different systems asking for torque.

Technical Breakdown & Logic: The calculation is based on **Priority Ranking**.

1. Multiple modules (Cruise Control, Traction Control, Idle Control, and the Driver) all send "Torque Requests" simultaneously.
2. The Coordinator looks at the "Status Bits" for each request.
3. For example, if the DSC (Traction Control) sends a request to *reduce* torque because the wheels are spinning, and the Driver is asking for *full* torque, the Coordinator chooses the **Minimum** of the two values to ensure safety.
4. The final result is the **Indicated Target Torque**, which is then passed down to the Injection and Ignition modules to be turned into physical engine work.

Ghidra Reference: Search for the function that integrates the variable `mi_soll` (Target Torque). This function is a massive junction point in the code. You will see inputs coming in from the CAN bus (DSC/EGS) and internal calculations (FGR/LLR). Following the output of this function will lead you directly to the maps that convert torque into `rl` (Relative Filling).

[Section 43] Emergency Running & Limp Home (Notlaufkonzepte)

When a critical sensor fails, the ECU must "guess" the values to keep the engine running safely.

Technical Breakdown & Logic: The calculation is a **Substitute Value Strategy**.

- If the **HFM** (Mass Air Flow sensor) fails, the ECU can no longer calculate air mass directly.
- It switches to a calculation called **Alpha-N**, which uses the Throttle Position (`dk_ist`) and Engine Speed (`n`) to estimate the air mass from a backup map.
- The ECU then sets a "Substitute Flag" bit, which usually limits the maximum RPM and disables high-performance features like VANOS to prevent engine damage under uncertainty.

Ghidra Reference: Look for conditional branches that check the status of error bits like `b_hfm_err`. You will see the code "branch" away from the standard calculation to a secondary

routine. This secondary routine uses a different set of maps (Emergency Maps) which are usually much more conservative than the "Base" maps found in `KF_ZW_GRUND`.

[Section 44] Transmission Interface & CAN Logic (Getriebeschnittstelle)

This module handles the high-speed data exchange between the DME and the Transmission Control Unit (TCU/GS19).

Technical Breakdown & Logic: The calculation is based on **Signal Synchronization**. The DME receives a "Torque Reduction Request" (`mi_red_req`) via the CAN bus.

1. The DME calculates the current lead time required to drop engine torque exactly when the clutches are transitioning.
2. If the SMG/DCT is shifting up, it sends a request to the DME to "blip" or "cut" torque.
3. The DME calculates the necessary ignition retard (`dzw_getr`) and fuel cut-off (`b_st_getr`) to achieve the torque drop within milliseconds. This is why these engines "pop" or "crack" during high-load shifts.

Ghidra Reference: Search for the CAN receive buffers, specifically looking for the message IDs associated with the **GS19 (SMG/DCT)**. Trace the variable `mi_getr` (Transmission Torque). You will find logic that overrides the driver's pedal request (`mi_fap`) during the shift phase.

[Section 45] SMG Shift-Point Logic / Rev Matching (20.2 Zwischengas-Funktion)

Specific to the S85/SMG setup, this module manages the "Blip" (Zwischengas) during downshifts.

Technical Breakdown & Logic: The calculation is a **Dynamic RPM Target**. When the SMG initiates a downshift, it sends a target RPM to the DME.

1. The DME calculates the "Delta RPM" (the gap between current speed and the target speed of the lower gear).
2. It triggers a "Positive Torque Intervention."
3. Instead of using the driver's foot position, the ECU uses the **Idle Actuators (LLS)** and **Throttles (DK)** to rapidly flare the RPM.
4. The calculation is finished when the `n_ist` (Actual RPM) matches the `n_getr_soll` (Transmission Target RPM) within a tolerance of `K_N_SYNC_TOL`.

[Section 46] DCT/DKG Shift Transitions (DKG-Übergangsschaltung)

Unlike the SMG, which is a single-clutch automated manual, the DCT requires a "Torque Cross-over" logic.

Technical Breakdown & Logic: The calculation is a **Torque Ramp**. During a DCT shift, Torque is moved from one clutch to the other.

1. The DME must maintain a perfectly steady "Virtual Torque" while the physical load shifts from Clutch A to Clutch B.
2. The logic uses a **Gradient Limitation** calculation.
3. If the torque is dropped too fast, the shift feels "jerky"; if it is too slow, the clutches overheat.
4. The DME uses the variable `mi_verlust` (Loss Torque) to account for the drag of the dual-clutch assembly when calculating how much "Indicated Torque" to produce.

Ghidra Reference: The DCT logic is more complex and involves "Torque Shaping." Look for references to `mi_ind_soll` (Indicated Target Torque). You will see complex 2D maps that determine how fast the engine can ramp torque back up after a shift is completed (`K_MI_GRAD_UP`).

[Section 47] Launch Control Logic (Launch-Control)

This is a coordinated effort between the DSC, the Transmission, and the DME.

Technical Breakdown & Logic: The calculation is an **RPM-Slip Control Loop**.

1. The DME receives a "Launch Active" bit (`b_lc_act`).
2. It ignores the pedal position and holds the engine at a steady RPM defined by the constant `K_N_LC`.
3. As the driver releases the brake/paddle, the DME calculates the maximum allowed wheel slip.
4. If the slip exceeds the target, the DME calculates an **Ignition Retard** to kill power instantly, rather than closing the throttles, because ignition retard is faster and keeps the manifold pressure high.

Ghidra Reference: Look for the function that handles `n_max_lc`. This is a secondary rev-limiter that is only active when the vehicle speed is 0 and the transmission is in a specific "Ready" state.

[Section 48] M-Drive & Torque Mapping M-Drive Logik)

The MS_S60 utilizes a "Torque-Based" driver demand system. The "Power" button on the console does not actually increase engine horsepower; instead, it changes the mathematical relationship between the pedal position and the throttle request.

Technical Breakdown & Logic: The calculation involves a **Pedal Map Selection**. The ECU monitors the status of the `b_mdrive` bit (received via the CAN bus from the M-Drive steering wheel button).

1. **Normal Mode:** The ECU uses map `KF_PED_NORM`. This map is "Linear" or "Progressive," meaning the first 50% of pedal travel only requests about 30% of engine torque. This makes the car easier to drive in traffic.
2. **Sport/Sport Plus Mode:** The ECU switches to `KF_PED_SPORT`. The logic here is "Aggressive." 50% of pedal travel might request 80% of available torque.
3. **Throttle Response:** In "Sport Plus," the ECU also bypasses certain **Gradient Limiters** (`K_MI_GRAD_MAX`), allowing the throttles to snap open faster.

Ghidra Reference: To find the M-Drive logic, search for the code that selects the index for the pedal-to-torque conversion. Look for a conditional branch (`if/else`) that checks a RAM variable tied to the M-Drive state. Trace this to the map lookup function for the relative filling (`rl`) request.

[Section 49] Power Management & Electrical Load (Energiemanagement)

Because the S65/S85 engines have high electrical demands (e.g., dual electric fans, ion-current modules, and electric oil pumps), the ECU must manage the "Electrical Torque" of the alternator.

Technical Breakdown & Logic: The calculation is based on **Alternator Drag Compensation**.

1. The alternator creates mechanical resistance on the crankshaft. The ECU calculates this as a "Loss Torque" (`mi_gen_verl`).
2. **The Logic:** If the battery is low, the alternator works harder, increasing `mi_gen_verl`.
3. To prevent the idle speed from dropping, the ECU automatically increases the **Idle Actuator (LLS)** position to compensate for the electrical drag.

4. **Calculated Variable:** `mi_ist = mi_combustion - mi_friction - mi_gen_verl`.

Ghidra Reference: Look for the **BSD (Bit-Serial Data)** communication drivers. This is where the ECU sends the target voltage to the alternator. You will see a calculation that takes the battery temperature (**`tbat`**) and outputs a voltage between 12.0V and 15.5V.

[Section 50] Final Torque Coordination (Momentengenerierung)

This is the final stage where all the requests (Driver, Cruise Control, SMG/DCT, and Stability Control) are flattened into a single physical instruction for the engine.

Technical Breakdown & Logic: The calculation is a **Hierarchy of Intervention**.

1. **Phase 1 (The Coordinator):** The ECU looks at all incoming `mi_soll` requests. It chooses the "Most Urgent" (usually the DSC/Traction Control for safety).
2. **Phase 2 (The Realization):** The ECU splits the final torque request into two paths:
    - **The "Slow" Path (Air):** The ECU adjusts the **Throttle Butterflies (DK)** and **Idle Actuators (LLS)**. This is slow because air has mass and takes time to move.
    - **The "Fast" Path (Ignition):** The ECU retards the **Ignition Angle (`zwout`)**. This is nearly instant (milliseconds) and is used for shift-cuts and traction control.
3. **The Result:** The engine produces the exact Newton-meters (**Nm**) requested by the drivetrain coordinator.

Ghidra Reference: This is the "Main Loop" of the ECU. Search for the function that converts **`mi_soll`** into **`rl`** (Relative Filling). This function is the "Holy Grail" for tuners because it controls the entire "feel" of the engine. It contains the inverse calculations of the engine's volumetric efficiency.

[Section 51] Main Fueling Model (Kraftstoffmodell)

The MS_S60 uses a "Physical Model" for fueling. It does not just look up a value in a table; it calculates the required fuel mass based on the ideal gas law and the target stoichiometric ratio.

Technical Breakdown & Logic: The calculation is based on the **Stoichiometric Equation**. The ECU's goal is to match the actual air mass with the exact amount of fuel needed to reach the target Lambda.

1. **Air Mass Calculation (ml):** The ECU calculates the mass of air entering the cylinder using the **HFM** (Mass Air Flow sensor) or the **Alpha-N** backup (Throttle position vs. RPM).
2. **Base Fuel Mass Calculation:** The ECU divides the air mass by the stoichiometric constant (14.7 for gasoline) and then multiplies by the **`lam_soll`** (Target Lambda).

3. **Wall Film Compensation (`wall_f`):** This is a critical "Transient" calculation. When you snap the throttle open, some fuel sticks to the walls of the intake port instead of entering the cylinder.
    - **Logic:** The ECU calculates the "puddle" size based on `tmot` (engine temp). It adds extra fuel during acceleration to compensate for the fuel sticking to the walls, and subtracts fuel during deceleration as that "puddle" evaporates into the engine.

Ghidra Reference: Search for the variable `mff_sk` (Fuel Mass). The wall film logic is often found in a function that uses constants like `K_WF_TAU` (time constant for evaporation) and `K_WF_GK` (filling constant). Tracing these will show you the "Transient Enrichment" logic.

[Section 52] Low-Pressure Fuel System (Kraftstoff-Niederdruck)

Unlike older systems with a mechanical regulator, the MS_S60 manages a **Demand-Controlled** fuel pump via the **EKP (Electronic Fuel Pump)** module.

Technical Breakdown & Logic: The calculation is a **Pressure-Flow Correlation**. The S65/S85 requires a constant differential pressure across the injectors.

1. **Target Pressure:** The ECU determines the target fuel pressure (typically 3.0 to 6.0 bar) based on the current fuel flow requirement.
2. **PWM Command:** The ECU sends a PWM signal via the CAN bus to the EKP module.
3. **The Calculation:** The ECU uses the map `KF_KLFP_PWM`, indexed by Battery Voltage and Required Flow. This reduces the load on the alternator and prevents the fuel from overheating by not pumping more than necessary.

Ghidra Reference: Search for the CAN transmission of the `EKP_Request`. The logic that calculates this request involves a lookup of the injector flow rate variables.

[Section 53] Full Load Enrichment (Vollfettanreicherung)

To protect the engine at high RPM and high load, the ECU must deviate from the economy-focused Lambda 1.0.

Technical Breakdown & Logic: The calculation is an **Exhaust Gas Temperature (EGT) Model**.

1. **Model Input:** The ECU does not have an EGT sensor; it *calculates* the temperature based on ignition timing, RPM, and load.
2. **Enrichment Trigger:** If the calculated temperature exceeds the limit `K_ABG_TEMP_MAX`, the ECU enters "Component Protection" mode.
3. **The Calculation:** The `lam_soll` is dropped (e.g., from 1.0 to 0.85). This extra fuel acts as a coolant for the exhaust valves and the catalytic converters.

Ghidra Reference: Look for the function that modifies `lam_soll`. You will see a comparison against a calculated temperature variable (often called `tabg_ist`). If you want to change the "Full Throttle" fueling, you must find the map `KF_LAM_VL` (Lambda Full Load).

[Section 54] Deceleration Fuel Cut-Off (Schubabschaltung)

To save fuel and provide engine braking, the ECU stops injection entirely when the driver lifts off the throttle.

Technical Breakdown & Logic: The calculation is a **Hysteresis Loop**.

1. **Cut-Off Condition:** If the pedal is at 0%, the RPM is above `K_N_SA_EIN`, and the engine is warm, the bit `b_sa` (Schubabschaltung) becomes True.
2. **Injection Pulse:** The pulse width `te` is set to 0.0ms.
3. **Re-activation:** To prevent the engine from stalling, the ECU must turn the fuel back on *before* it reaches idle speed. This is defined by the constant `K_N_SA_AUS` (typically around 1,200 RPM).

Ghidra Reference: Search for the bit `b_sa`. The logic surrounding this bit contains the timers for the "Exhaust Burble" or "Pop and Bang" settings. By delaying the transition of `b_sa` and maintaining a small injection pulse with retarded ignition, tuners create the overrun sound effects.

[Section 55] Torque Manager: Structure & Arbitration (Momentenmanager)

The Moment Manager doesn't just calculate torque; it arbitrates between conflicting "Torque Desires" from various systems.

Technical Breakdown & Logic: The calculation follows a **Hierarchical Priority Logic**. It takes dozens of inputs and filters them through a "Min/Max Selection" process.

1. **Internal Requests:** Idle speed control (`mi_llr`), engine friction compensation (`mi_fric`), and component protection.
2. **External Requests:** Driver's pedal (`mi_fap`), Transmission (DCT/SMG) shifts (`mi_getr`), and Stability Control (`mi_dsc`).
3. **The Arbitration (Selection):**
    - The ECU first determines the **Permissible Torque (`mi_zul`)**.
    - It then calculates the **Target Indicated Torque (`mi_ind_soll`)**.
    - **Logic:** If the DSC requests 200Nm to stop wheelspin, but the Driver requests 500Nm, the Moment Manager selects the **Minimum** of the two. If the Idle

Controller requests 50Nm to prevent a stall, but the friction model says the engine needs 60Nm just to stay turning, it selects the **Maximum**.

Ghidra Reference: Search for the variable **`mi_ind_soll`**. This is the "Master Variable." Trace it backward to find the **`m_sel`** (Moment Selection) function. This function is a massive block of `if-else` statements that compare every active torque request.

[Section 56] Torque Realization: The Two Paths (Momentenumsetzung)

Once the Moment Manager decides on a final Torque value, it must decide *how* to make the engine produce it. It splits the request into a **Slow Path** and a **Fast Path**.

Technical Breakdown & Logic: The calculation is an **Efficiency-Based Split**.

1. **The Slow Path (Air Path):**
    - **Target:** Adjusts the **Throttle Butterflies (DK)** and **Idle Actuators (LLS)**.
    - **Logic:** This path is used for long-term torque changes (like cruising). The ECU calculates the required **Relative Filling (`rl_soll`)** to achieve the torque.
2. **The Fast Path (Ignition/Lambda Path):**
    - **Target:** Adjusts the **Ignition Angle (`zwout`)** or cuts fuel.
    - **Logic:** This path is used for instant changes (like a DCT shift-cut or Traction Control).
    - **Calculation:** The ECU calculates an **Efficiency Factor (`etazw`)**. If the Manager needs to drop torque *faster* than the throttles can move, it retards the ignition, effectively "wasting" energy to meet the torque target instantly.

Ghidra Reference: Find the function that calculates **`etazw`** (Ignition Efficiency). You will see it compares the **`zwout`** (actual) to **`zwopt`** (optimal). If `etazw` is less than 1.0, the Moment Manager is intentionally killing power to meet a fast-path request.

[Section 57] Drag Torque & Friction Model (Schleppmoment / Reibmodell)

To be accurate, the Moment Manager must know how much torque is lost just by the engine turning itself.

Technical Breakdown & Logic: The calculation is a **Loss Summation**. The ECU calculates the "Internal Friction Torque" (**`mi_fric`**) which is the force required to move the pistons, oil pump, and valvetrain.

1. **The Formula:** `mi_loss = mi_fric(n, tmot) + mi_acc(alternator/AC)`.

2. **Logic:** As the engine oil gets warmer (`tmot`), the friction decreases. The Moment Manager subtracts this loss from the "Indicated Torque" (what happens at the spark plug) to find the "Brake Torque" (what actually reaches the flywheel).

Ghidra Reference: Look for the map `KF_MI_FRIC`. This is a 16x16 map indexed by RPM (`n`) and Oil/Coolant Temp. Tuning this map is necessary if you install a lightweight flywheel or high-flow oil pumps, as the ECU's "internal math" for idle stability will otherwise be wrong.

[Section 58] Monitoring Level 2: Torque Plausibility (Momentenüberwachung)

This is the safety "Watchdog" that prevents unintended acceleration.

Technical Breakdown & Logic: The calculation is a **Redundant Cross-Check**. A separate part of the CPU (Level 2) calculates the torque independently of the Main Manager (Level 1).

1. **The Logic:** Level 2 calculates the maximum "Allowed Torque" (`mi_zul_mon`) based strictly on the driver's pedal position.
2. **The Comparison:** If the Main Manager's **Actual Torque** exceeds the Level 2 **Allowed Torque**, the ECU assumes a software crash or hardware failure.
3. **The Action:** The ECU triggers `b_st_enz` (Injection Cut-off) and sets the limp-home mode.

Ghidra Reference: This is often found in the "Safety Monitoring" sector of the binary. Search for the address of `K_MI_MAX_ERR`. If you are building a high-boost or supercharged S65/S85, you often have to increase these monitoring thresholds in Ghidra/Hex, or the car will "Limp Home" the moment it sees more torque than the factory allowed.

[Section 59] PT-Commands: Injector Driver Control (Einspritzendstufen - PT)

While the Fueling Model calculates the fuel mass, the **PT-Command** handles the electrical firing of the injector.

Technical Breakdown & Logic: The calculation is a **Time-to-Angle Conversion**. The CPU works in "Time" (milliseconds), but the engine works in "Angle" (degrees Crankshaft).

1. The PT-Command takes the calculated injection time (te) and the current engine speed (n).
2. It calculates the exact hardware timer "ticks" required to open the injector at the start angle (`asoe`) and keep it open for the duration of te.
3. **Voltage Correction:** The PT-command adds the battery voltage compensation from `KL_TE_UB` at the very last microsecond before firing to ensure accuracy.

Ghidra & PT Command: Look for calls to the **TPU (Time Processing Unit)** or **eTPU** functions. The PT command for injection is often a specific function call that passes three parameters: `Cylinder_ID`, `Start_Angle`, and `Duration_Ticks`.

[Section 60] PT-Commands: Ignition Triggering (Zündendstufen - PT)

The ignition PT-command manages the "Dwell Time" (the time the coil is charged) and the "Spark Event."

Technical Breakdown & Logic: The calculation is based on **Dwell Mapping**.

1. The logic determines the **Dwell Time** (the "Charge" phase) based on battery voltage. A low battery requires a longer charge time to create a strong spark.
2. The PT-command subtracts the Dwell Time from the target ignition angle (**`zwout`**) to find the "Charge Start" point.
3. **Execution:** It triggers the high-side driver to begin charging the coil, then abruptly cuts the ground to collapse the magnetic field and fire the spark exactly at `zwout`.

Ghidra & PT Command: Search for the constant **`K_T_ZUE_LADE`** (Ignition Charge Time). The PT code here is extremely time-sensitive and usually resides in the highest-priority interrupt service routine (ISR).

[Section 61] PT-Commands: PWM Actuator Drivers (26.3 PWM-Ansteuerung - PT)

For components like the **VANOS solenoids**, **Idle Actuators (LLS)**, and **Electric Oil Pumps**, the PT-commands translate a 0–100% request into a Pulse Width Modulated signal.

Technical Breakdown & Logic: The calculation is a **Duty Cycle Modulation**.

1. The PT-command receives a percentage (e.g., 45% for a VANOS solenoid).
2. It calculates the "On-Time" vs. "Off-Time" based on the fixed carrier frequency of that specific component (e.g., 300Hz for VANOS).
3. **Dither Logic:** To prevent the VANOS solenoids from sticking (hysteresis), the PT-command often adds a small "Dither" or vibration signal to the PWM output, keeping the mechanical valve in a state of constant, micro-movement.

Ghidra & PT Command: Search for the hardware register addresses for the **PWM Channels**. In the MS_S60, these are often memory-mapped I/O. Tracing the "Write" operations to these registers will lead you to the PT-layer code that converts the logical setpoints (like `van_soll`) into electrical signals.

[Section 62] PT-Commands: CAN Bus Communication (CAN-Bus - PT)

This is the PT-command for the "Postal Service" of the ECU. It packages internal variables into 8-byte frames for the DCT, SMG, and DSC.

Technical Breakdown & Logic: The calculation is **Byte-Packing and Scaling**.

1. Internal ECU variables (like Torque) are stored as 16-bit or 32-bit values with high precision.
2. The PT-command scales these values to fit into the standard CAN frame (e.g., a 0–1000Nm torque value might be scaled by 0.5 to fit into a single byte).
3. **Checksum/Alive Counter:** For safety-critical messages (like the Torque Request from the DCT), the PT-command adds a "Rolling Counter" and a Checksum to the end of the frame. If the receiver sees a gap in the counter, it ignores the message.

Ghidra & PT Command: Look for the **CAN Mailbox** configurations. The PT-commands here are responsible for moving data from the RAM addresses (like `mi_ist`) into the CAN Controller's transmit registers.

[Section 63] Secondary Air Mass Meter (3.2 mHF - Sekundärluft-Heißfilm-Luftmassenmesser)

The Secondary Air Injection (SLP) system has its own dedicated mass air meter to verify the pump is actually moving air into the exhaust.

Technical Breakdown & Logic: The calculation is a **Flow Comparison**.

1. **Measured Flow (`m_slp_ist`)**: The mHF sensor sends a voltage signal to the ECU representing the fresh air being pumped into the exhaust.
2. **Modeled Flow (`m_slp_mod`)**: The ECU calculates what the flow *should* be based on the SLP pump speed and system voltage.
3. **The Calculation**: The ECU compares the measured vs. modeled flow. If the deviation exceeds the threshold `K_SLP_MAX_DIFF`, the ECU sets the bit `b_slp_err`. This ensures that a clogged air valve or a failing pump is detected for emissions compliance.

Ghidra & PT-Command: The mHF sensor is read via an Analog-to-Digital Converter (ADC) channel. Look for the PT-command that converts the raw voltage into a mass flow value using the linearization curve `KL_SLP_ML`.

[Section 64] Manifold Pressure Sensors (Saugrohrdruckfühler - MAP)

The S65 and S85 utilize two MAP sensors (one for each plenum/bank). Because these engines use individual throttle bodies (ITBs), the MAP signal is extremely "pulsatile" and requires heavy filtering.

Technical Breakdown & Logic: The calculation is based on **Pressure-Model Synchronization**.

1. **Filtering (`p_intake_filt`)**: Because the pressure fluctuates every time a valve opens, the ECU uses a "Windowed Average" to find the mean pressure in the plenum.
2. **Plausibility Check**: The ECU compares the measured pressure (`ps[j]`) against the calculated air mass from the **HFM**.
3. **The Logic**: Under steady-state conditions, the pressure must match the air mass and throttle position. If ps[j] deviates from the modeled pressure by more than `K_P_DIFF_MAX`, the ECU assumes a vacuum leak or a sensor failure.

Ghidra Reference: Search for the variables `ps1` and `ps2` (Pressure Sensor Bank 1 and 2). Tracing these will lead you to the **Ambient Pressure Calibration** routine, where the ECU updates the atmospheric pressure constant (`p_amb`) every time the engine is at wide-open throttle or before the start.

[Section 65] Emergency Load Calculation (Ersatzlast / Alpha-N)

If the HFM or MAP sensors fail, the ECU must calculate load using the "Alpha-N" method (Throttle Angle + Engine Speed).

Technical Breakdown & Logic: The calculation is a **Volumetric Efficiency (VE) Lookup**.

1. **Inputs**: The ECU takes the throttle angle (**α**) and engine speed (**n**).
2. **The Calculation**: It looks up the base air filling in map `KF_RL_ALPHAN`.
3. **Corrections**: This base value is corrected by the intake air temperature (`tans`) and the manifold pressure (if the sensor is still working). If the MAP is also dead, the ECU uses the atmospheric pressure constant `p_amb`.

Ghidra & PT-Command: Search for the bit `b_hfm_fail`. When this bit is set, the code branches to the Alpha-N routine. You will see a PT-command that switches the primary load variable `rl` from the HFM input to the output of the `KF_RL_ALPHAN` map.

[Section 66] Electric Cooling Fan Control (Elektrolüftersteuerung)

The electric fan on the MS_S60 is a high-output PWM-controlled unit. It is responsible for thermal management of the coolant, the engine oil (via the oil-to-air cooler), and the A/C condenser.

Technical Breakdown & Logic: The calculation is a **Multi-Input Maximum Selection**. The ECU does not look at just one temperature; it evaluates several "heat sources" and chooses the highest required fan speed to satisfy the most demanding system.

1. **Coolant Demand (`n_fan_tmot`)**: Calculated from a map based on the radiator outlet temperature. If the coolant leaving the radiator is too hot, the fan speed increases.
2. **A/C Pressure Demand (`n_fan_ac`)**: The A/C module (IHKA) sends a request via CAN based on refrigerant pressure. High pressure requires more airflow across the condenser.
3. **Oil Temperature Demand (`n_fan_toel`)**: Unique to the S65/S85, if the engine oil temperature exceeds a threshold, the fan is ramped up to pull air through the dedicated oil cooler.
4. **The Calculation**:
    - `n_fan_soll = MAX(n_fan_tmot, n_fan_ac, n_fan_toel)`
    - The ECU then applies a **Hysteresis** to prevent the fan from rapidly cycling on and off.
    - The final value is converted to a PWM duty cycle (typically 10% to 90%).

Ghidra & PT-Command: The fan is controlled via a single-wire PWM output. Search for the variable `pwm_fan`. The PT-command translates the `n_fan_soll` into a hardware timer signal. Look for the constant `K_FAN_OFF_TEMP`; if you change this, you change the temperature at which the fan first kicks in.

[Section 67] Electric Water Pump / After-Run (Nachlaufsteuerung)

While the main water pump on the S65/S85 is belt-driven, some variants and specific cooling circuits use an auxiliary electric pump to prevent "Heat Soak" after the engine is turned off.

Technical Breakdown & Logic: The calculation is an **Energy Accumulation Model**.

1. The ECU monitors the engine temperature and the "Heat Input" (how hard the car was driven) just before shutdown.
2. **After-Run Requirement**: If the coolant temperature at shutdown is above `K_TMOT_NACH_LIM`, the ECU enters "After-Run" mode.
3. **The Action**: The ECU keeps the electric fan and the auxiliary water pump running for a duration calculated by the map `KF_T_NACHLAUF`. This prevents the coolant from boiling inside the cylinder head after the mechanical pump stops.

[Section 68] Alternator Load Management (Generator-Momenten-Schnittstelle)

Because the electric fan can pull over 60 Amps, the Torque Manager must be notified whenever the fan ramps up.

Technical Breakdown & Logic: The calculation is a **Predictive Torque Offset**.

1. Before the PT-command increases the fan's PWM duty cycle, it sends a signal to the **Moment Manager**.
2. The Moment Manager calculates the extra mechanical drag the alternator will place on the crankshaft to provide that 60A.
3. The **Idle Actuators (LLS)** are opened slightly *before* the fan starts to ensure the RPM doesn't "dip" when the electrical load hits.

Ghidra Reference: Trace the variable `mi_gen_soll`. You will see it is the sum of various electrical consumers. The electric fan's contribution is scaled by the constant `K_FAN_PWR_TO_NM` (a conversion factor from fan power to Newton-meters of engine torque).

[Section 69] PT-CAN Address Map & Signal Decoding (CAN-Botschaften)

The MS_S60 uses a "Mask and Filter" hardware logic to capture specific Hexadecimal IDs from the bus.**Technical Breakdown & Logic:** The calculation involves **Endianness and Scaling**.

- **The Logic:** Most signals are sent as "Motorola" (Big Endian) format. The PT-command must "byte-swap" the raw data before the application logic can use it.
- **Scaling Example:** The engine speed (`n_ist`) is often sent with a factor of 0.25. If the raw hex value is `0x4E20` (20,000 decimal), the PT-command calculates:
  $20{,}000 \times 0.25 = 5{,}000$ RPM

[Section 70] Torque Intervention Signals (Momenteneingriff PT-Layer)

When the DCT/SMG or DSC wants to control the engine, it sends specific "Intervention" signals that the Moment Manager must prioritize.

Key Signal Names:

Hex ID	Message Name	Source	Key Signals / Variables
0x0A8	TORQUE_1	DME	mi_ist (Actual), mi_ind (Indicated)
0x0AA	RPM_SPEED	DME	n_ist (RPM), v_ist (Wheel Speed)

0x0B6	GEAR_S	GS19	curr_gear (Actual Gear), tar_gear (Target Gear)
0x0D0	DSC_1	DSC	b_bremse (Brake), m_red_dsc (Torque Red.)
0x1D0	ENGINE_DATA	DME	tmot (Coolant), toel (Oil), p_amb (Pressure)
0x315	VEH_CONF	CAS	c_variant (Coding), vin_code (VIN)
0x545	CLUSTER_1	DME	b_check_engine (MIL), f_consumption

- `M_RED_GEAR` **(Torque Reduction Gear):** Used during upshifts to drop torque for clutch protection.
- `M_VERL_GEAR` **(Loss Torque Gear):** The drag created by the dual-clutch or SMG assembly.
- `B_LC_READY` **(Launch Control Ready):** A bit sent by the gearbox to tell the DME to activate the Launch RPM limiter.

Technical Breakdown & Logic: The PT-command processes these using a **Timeout Watchdog**.

1. If the message `0x0B6` (from the gearbox) is not received within `K_CAN_TIMEOUT` (typically 100ms), the PT-layer sets the bit `b_can_err_gs`.
2. **Safety Action:** The Moment Manager immediately ignores all torque requests from that ID and reverts to the "Driver Only" torque map (`KF_PED_NORM`) to prevent the car from being stuck in a torque-reduction state.

[Section 71] Instrument Cluster & User Interface (Kombi-Instrument PT-Logik)

The DME controls the "Shift Lights" and the "Variable Redline" on the S65/S85 clusters.

Technical Breakdown & Logic: The calculation is a **Temperature-to-RPM Limit Mapping**.

1. The DME calculates the safe maximum RPM (`n_max_warm`) based on `toel` (Oil Temperature).

2. **PT-Command:** This value is packed into message `0x1D0` and sent to the Cluster (KOMBI).
3. **Shift Lights:** As the actual RPM (`n_ist`) approaches the limit, the PT-command triggers the "Yellow" and "Red" LED bits in the CAN message to warn the driver.

Ghidra Reference: To modify the shift light behavior or the variable redline, search for the function that builds message `0x1D0`. Look for a lookup table that maps `toel` to a 16-bit RPM value. Tracing this in Ghidra will show you exactly how the DME communicates "Engine Readiness" to the driver.

[Section 72] PT-CAN Diagnosis (Bus-Check & Error States)

The ECU monitors the health of the entire network.

Technical Breakdown & Logic: The calculation is a **Bus-Load Monitor**.

1. The PT-layer counts the number of "Error Frames" on the physical wire.
2. If the error count exceeds `K_CAN_ERR_MAX`, the hardware enters a "Bus-Off" state to prevent it from crashing the rest of the car's network.
3. **Recovery:** The PT-command waits for a specific duration (`K_CAN_RECOVERY_T`) before attempting to re-join the bus.

Ghidra Reference: The CAN error handling is usually located in the **Hardware Abstraction Layer (HAL)** of the binary. Look for code that accesses the CAN Controller's status registers (e.g., `CAN_ESR`).

In the MS_S60 architecture, **EGAS** (Elektronisches Gaspedal) and **AQ** (Analoge Quittierung / Signal Acquisition) represent the critical safety bridge between the driver's physical foot movement and the digital torque request.

Because the S65 and S85 use individual throttle bodies (ITBs) that are physically disconnected from the pedal, these systems must ensure that a sensor failure never leads to unintended acceleration.

[Section 73] EGAS: Electronic Accelerator Pedal (Fahrpedalgeber)

The EGAS module translates the physical pedal position into a percentage value (0.0% to 100.0%) that the Torque Manager can understand.

Technical Breakdown & Logic: The calculation is based on **Double Redundancy**. The pedal assembly contains two separate potentiometers (Hall sensors) that move in opposite directions or at different voltage scales.

1. **Signal Decoding**: The ECU reads `pwg_1` and `pwg_2`.

2. **Cross-Check (Plausibility)**: The logic calculates a "Difference Value." If $|pwg\_1 - pwg\_2| > K\_PWG\_TOL$, the ECU assumes a sensor fault.
3. **Limp Mode**: If one sensor fails, the ECU limits the maximum throttle opening to approximately 25% (`K_EGAS_EMERGENCY`). If both fail, the engine is forced to a high idle of 1,200 RPM, and throttle response is disabled.

Ghidra & PT-Command: The PT-command for EGAS involves two ADC (Analog-to-Digital) channels. Search for the variables `pwg_w` (Pedal Angle). You will find a 2x2 or 1x16 characteristic curve `KL_PWG_LIN` that linearizes the raw voltage into a percentage.

[Section 74] AQ: Signal Acquisition & Quality (Analoge Quittierung)

AQ refers to the hardware and software layer responsible for the high-speed acquisition of analog signals and their "Acknowledgment" (verification) before they are used in the main logic.

Technical Breakdown & Logic: The calculation involves **Signal Conditioning and Buffering**. Before a raw voltage becomes a variable like `tmot` or `p_oel`, it passes through the AQ layer.

1. **Over-Sampling**: The AQ PT-command reads the sensor voltage 8 or 16 times in a single millisecond and calculates a **Mean Average**. This filters out electrical noise from the ignition coils.
2. **Range Check (Lower/Upper)**:
    - If the voltage is <0.2V, the AQ sets a "Short to Ground" bit (`b_aq_min`).
    - If the voltage is >4.8V, it sets a "Short to B+" bit (`b_aq_max`).
3. **Substitution**: If a bit is set, the AQ layer "Acknowledges" the failure and provides a substitute value (e.g., assuming 80°C for coolant if the sensor fails) so the main program doesn't crash.

Ghidra Reference: The AQ logic is usually found in a central "Input Manager" function. Look for a large table of constants that define the **Min/Max Voltage** thresholds for every sensor on the engine. Tracing the output of the ADC registers (like `ADC_RESULT_REG`) will lead you directly into the AQ filtering code.

[Section 75] EGAS-AQ Torque Monitoring (Sicherheitsüberwachung)

This is the "Level 2" monitoring that connects EGAS and AQ. It ensures the throttles (**DK**) actually followed the pedal (**PWG**).

Technical Breakdown & Logic: The calculation is a **Comparison of Desired vs. Actual Area**.

1. The ECU calculates the "Effective Flow Area" requested by the driver's pedal.
2. The AQ layer reports the "Actual Flow Area" based on the feedback from the throttle position sensors (`dk_ist`).
3. **The Safety Check**: If the Actual Area is significantly larger than the Desired Area for more than 200ms, the **Safety Path** triggers an immediate fuel cut-off (`b_st_enz`).

PT-CAN Signal Name:

- **PWG_CAN**: The pedal position sent to the Transmission and DSC.
- **DK_SOLL_CAN**: The throttle request sent to the secondary slaves.