Approaches to Embed a Software-based Adaptive Prognostic Estimation Kernel into a PHM System-on-Chip

James Hofmeister
Ridgetop Group, Inc.
3580 West Ina Road
Tucson, AZ 86741
520-742-3300x107
jhofmeister@ridgetopgroup.com

Wyatt Pena
Ridgetop Group, Inc.
3580 West Ina Road
Tucson, AZ 86741
520-742-3300x702
wpena@ridgetopgroup.com

Christopher Curti
Ridgetop Group, Inc. 3580
West Ina Road
Tucson, AZ 86741
520-742-3300x704
ccurti@ridgetopgroup.com

Abstract—This paper presents considerations and approaches related to embedding a software-based Adaptive Prognostic Estimation kernel into a Prognostic Health Management System-on-Chip (PHM SoC). Ridgetop Group's Adaptive Remaining Useful Life Estimation™ (ARULE™) software suite includes a two-stage Adaptive Prognostic Estimation kernel comprising robust, accurate, and a fast-converging predictive analytics kernel to produce meaningful prognostic estimates for State-of-Health (SoH), Remaining Useful Life (RUL), and Prognostic Horizon (PH). Existing PHM Standards and practices typically require a combination of hardware, firmware, and software elements for Sensors (S), Data Acquisition (DA), Data Management (DM), State Detection (SD), Health Assessment (HA), Prognostic Assessment (PA), Advisory Generation (AG), and Health Management (HM). For many PHM applications it is impractical to have a single SoC comprising all of the described PHM elements. We believe a more feasible and practical approach is to provide a PHM SoC solution having a multiplicity of sensors, microcontrollers, and a single Adaptive Prognostic Estimation SoC. The approach separates each PHM element into four overlapped groups: (1) S, DA, and DM; (2) DM, SD, and HA; (3) HA, PA, and AG; (4) AG and HM. Group 1 produces Feature Data (FD) that is extracted from Condition-Based Data (CBD) and has a characteristic signature that is highly correlated to degradation leading to failure. Groups 2 and 3 are a realization of group 2 and 3 as a single, two-stage SoC solution. Prognostication of a commercial-off-the-shelf quadcopter is used as an example to demonstrate a design of a PHM SoC in which AG is through light-emitting diodes (LED) and HM is initiated by an operator. A summary, conclusion, and follow-on activity section ends the paper.

Keywords—SoC, RUL, SoH, PH, predictive, prognostics, PHM, APK

TABLE OF CONTENTS

1. INTRODUCTION

This paper presents approaches to embed a software-based Adaptive Prognostic Estimation (APK) kernel (Figure 1) into a Prognostic Health Management (PHM) system-on-chip (SoC). As an initial assessment this "seems straightforward:" port software, with some modifications, into a microprocessor on a printed circuit board (PCB) and test to verify and validate – the need to design and develop at least one testbench (TB) for the PHM SoC.

We started with the IEEE 1856-2017 Standard Framework for Prognostics and Health Management of Electronic Systems and compared it to the envisioned PHM SoC as illustrated in Figure 2 and Figure 3. We then mapped Figure 3 to an existing hardware-software solutions (Figure 4). Figure 5 is a block diagram of an application program called UD_ARULE™ (User Data for Adaptive Remaining Useful Life Estimator™) that interfaces with a realized APK: the object module called ARULE.

Figure 4 and Figure 5 were used as the foundation for developing approaches to realize a PHM SoC as a multiple chip solution to achieve the following goals:

- Successful creation of a software version of ARULE for loading as firmware into a microprocessor: ARULE-on-chip with suitable functionality for PHM SoC solutions (right-hand side of Figure 5).
- Successful design and development of a software-based testbench (TB) to verify and validate a firmware version of ARULE and its interface to other chips.
- Successful export of ARULE as an APK into a microprocessor chip to support PHM SoC.
- Successful design and development of hardware-based TB to verify and validate the ARULE-on-chip solution.

978-1-6654-9032-0/23/$31.00 ©2023 IEEE

- Successful export of the TB(s) to a realized demonstration of a prognostic-enabled, handheld quadcopter.

This paper describes the following approaches:

- Create a simplified version of ARULE with suitable functionality for a chip-based solution.
- Replace file-based-definition methods with methods more suitable for the PHM SoC to include the following: support for system configurability, extendibility, and customization; support for checkpoint/restart mode of operation; and support for data acquisition and output.
- Eliminate current messaging and logging support, and improve support for health-related alerts.
- Design, develop, and perform TBs to verify and validate approaches and solutions,
- Design, develop, and demonstrate a proof of concept for a PHM SoC solution that uses ARULE-on-chip.

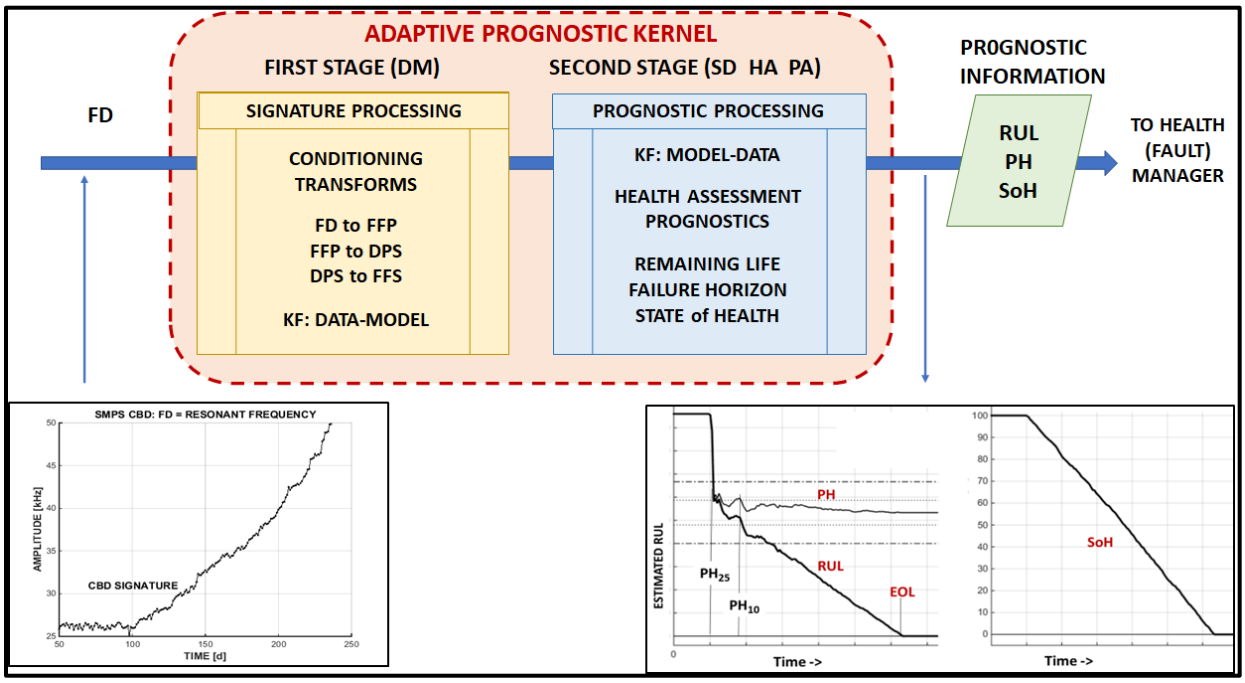


Figure 1. Block diagram of an Adaptive Prognostic Kernel (APK) comprising two stages, each of which uses a form of Kalman filtering

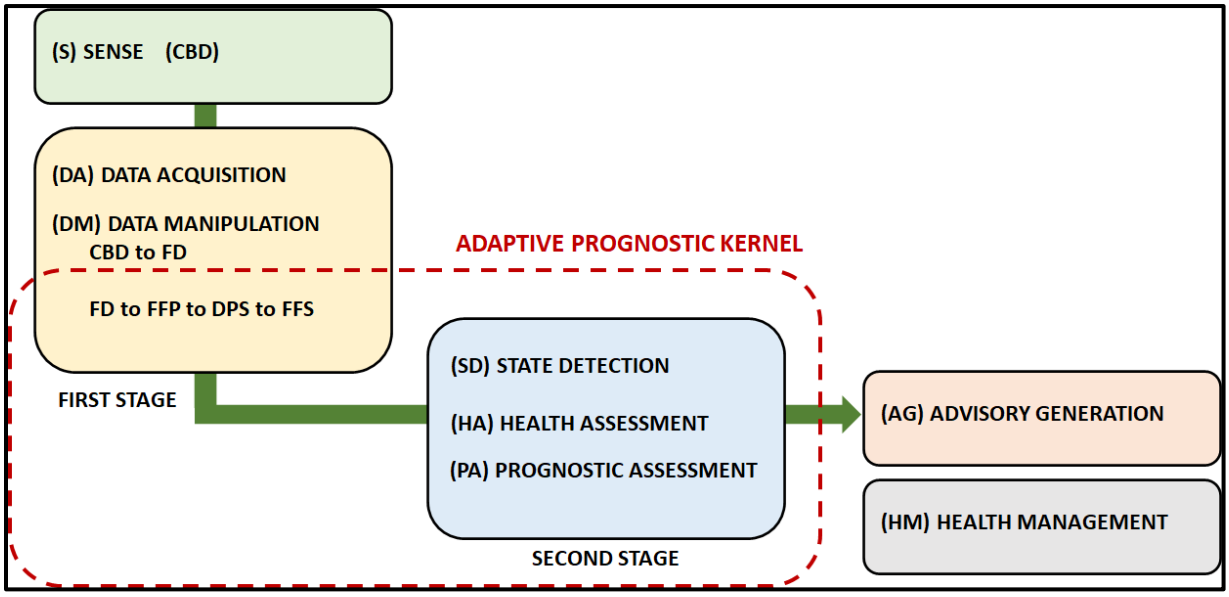


Figure 2. Relationship: IEEE 1856-2017 to the example APK shown in Figure 1

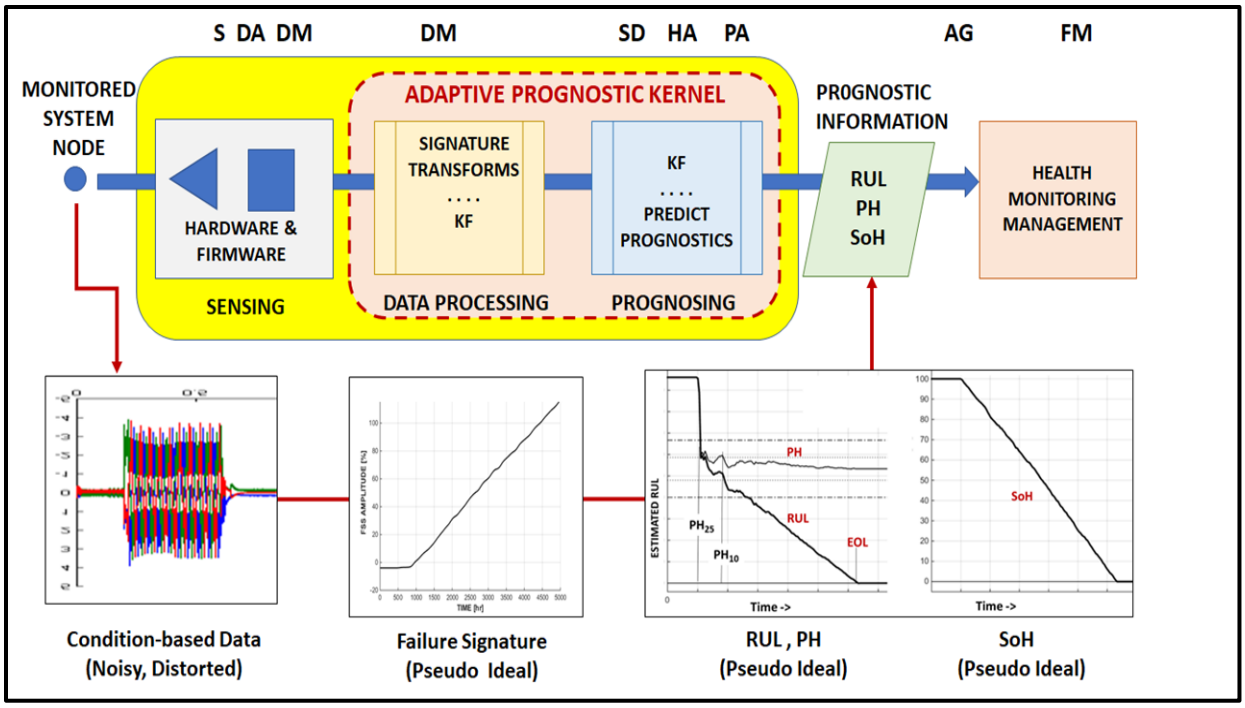


Figure 3. Block diagram representation of a realized, IEEE 1856-2017 compliant PHM system using a two-stage APK

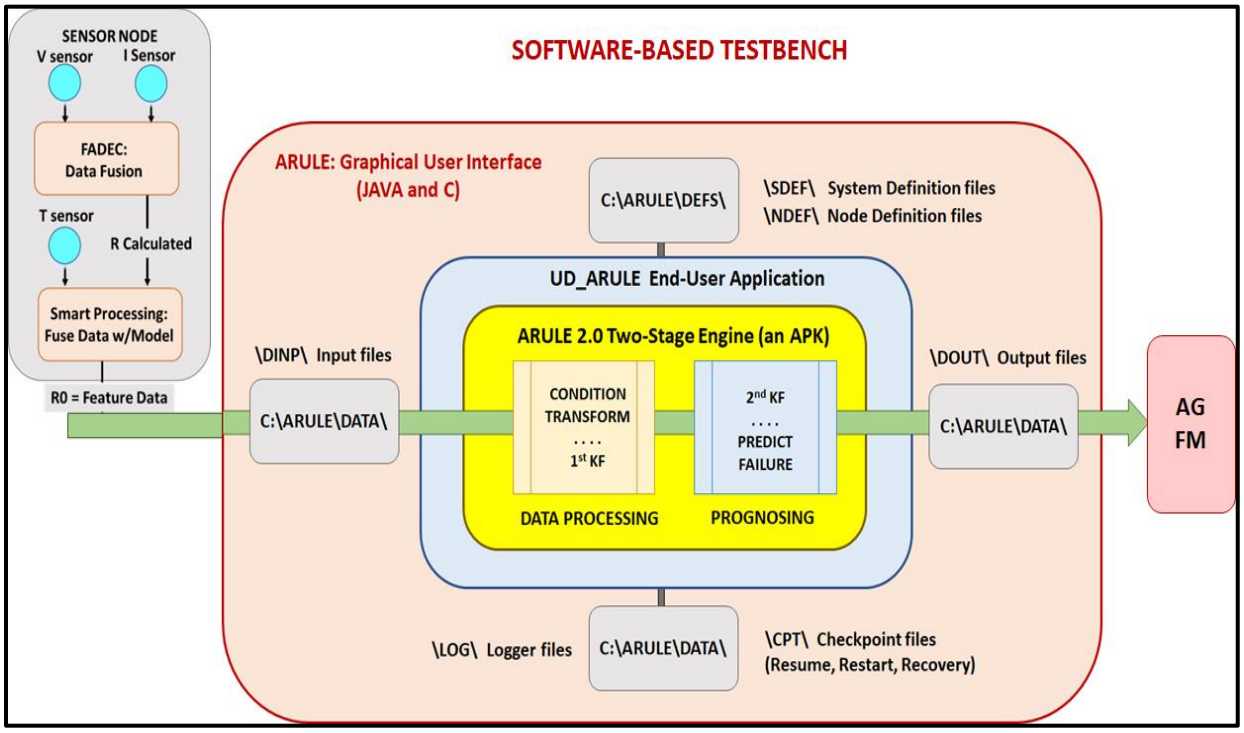


Figure 4. Block diagram: software-based testbench with UD_ARULE™ using an ARULE version of an APK

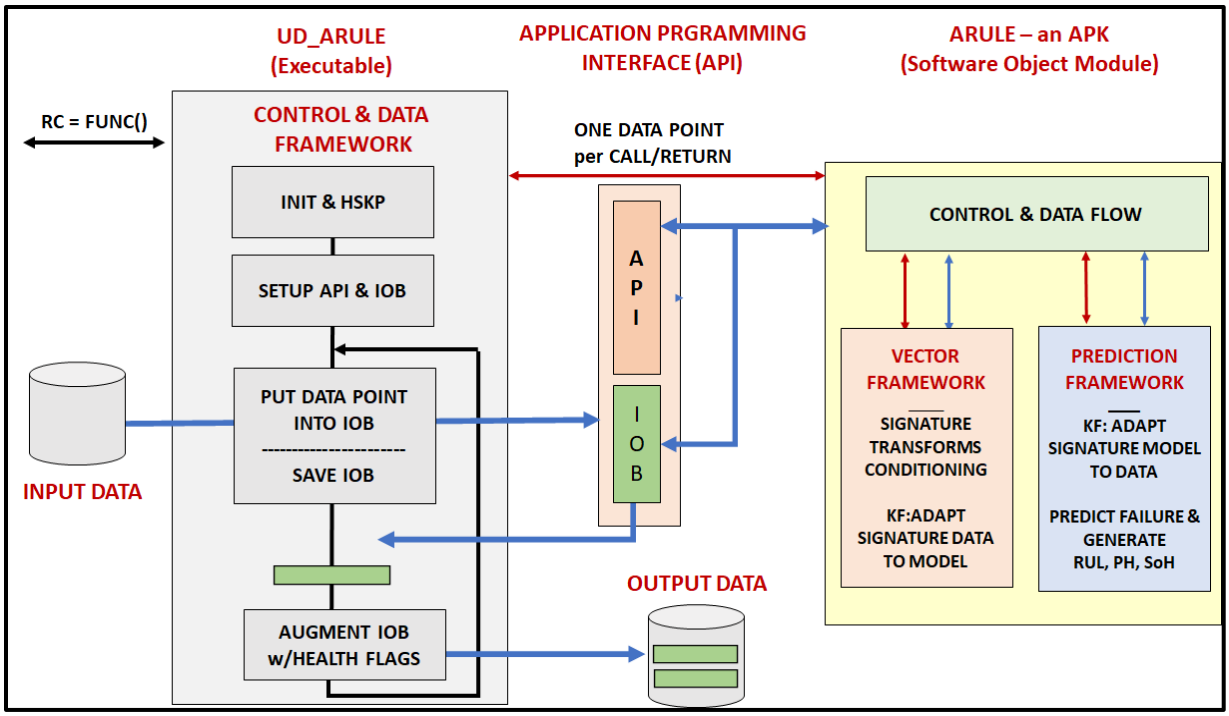


Figure 5. Block diagram: software-application program (executable) with embedded APK (object module)

2. Approach: Simplify ARULE APK

Software solutions are often replete with optional processing such as do A then B or do A then C. Our ARULE example of an APK, started out going directly from input feature data (FD) to Kalman Filtering (KF). Then the ability to transform FD signature data to Fault-to-Failure Progression (FFP) signature data was added to simplify data modeling followed by the ability to transform FFP signature data to Functional Failure Signature (FFS) data to simplify KF modeling; then transforming FFP data to Degradation Progression Signature (DPS) data to linearize curvilinear data to improve accuracy was added; and then KF was extended by adding a FFS data modeling step before the final KF and prediction processing. In addition to supporting selective sequencing, ARULE supports data smoothing of FD and/or FFP and/or FFS signatures. ARULE supports many input parameters and values that are not essential for producing accurate estimates of prognostic information.

We presented at this conference a paper that showed the feasibility of using a single family, rather than seven families, of DPS models; a single, optional data smoothing (FD); and fixed sequence of data and EKF processing: "Reducing Signature Models for Extended Kalman Filtering for Adaptive Prognostic Estimation" – see Figure 6 and Figure 7. The number of input parameters to control the operation of ARULE is reduced from 19 to nine and an additional 12 parameters used to support plotting are eliminated.

3. Approach: Replace File-based Methods more Suitable for PHM SoC

The ARULE example of an APK uses file-based methods to support the following: (1) definition files to configure, extend, and customize systems; (2) checkpoint/restart support; and (3) data acquisition and output. Those methods need to be replaced with methods suitable for PHM SoC – specifically, no reading or writing of direct-access storage devices (DASD).

A. System and Definitions

Readable and editable system and node definition files in c:\ARULE\DEFS\SDEF\ and c:\ARULE\DEFS\NDEF\ directories (Figure 4) are used to specify how systems are configured and how their nodes are processed. A method more suitable for PHM SoC is to include defined structures in a firmware APK along with an input to select which system instantiation applies.

Given a simplified version of ARULE as previously described, create structures equivalent to solution definitions that are embedded: no reading, no parsing, no interpretation, and no checking syntax, data type, or range of values.

```
%**Feature Data
FDNM       = 5.0;     % Noise margin
FDZ        = 24.0;    % Feature Data (FD)
FDZVAR     = 5.0;     % FD Nominal variance
FDZDIV     = 0;       % Ratio divisor
FDZPRE     = 0.0;     % Precision: 0/1 subtract/divide
%**Data Conditioning:
FDZPTS     = 10;      % # data points to average
FDPTS      = 5;       % Rolling FD (CBD)
FFPPTS     = 0;       % Rolling FFP points
FFSPTS     = 4;       % Rolling FFS points
FFSADJ     = 1;       % 0/1  no./yes
%**Degradation Signature - from PoF, FMEA
FFPFAIL    = 70.0;    % Failure margin - percent above nominal
XDPS       = 2;       % Power set 2
%  0 none    1 FD = FD0*(dP/P0)^n
%  2 FD = FD0*([1/(1 - dP/P0)]^n - 1)
%  3 FD = FD0*(1-[1/(1 + dP/P0)]^n)
%  4 FD = FD0*((1 + dP/P0)^n - 1)
%  5 FD = FD0*(1 - (1 - dP/P0)^n
%  6 FD = FD0*(exp(dP/P0) - 1)
%  7 FD = FD0*(1 - exp(-dP/P0))
XDPSNV     = 1.2;     % n or lambda
%**Prognostic Information
PITTFF     = 200.0;   % Default RUL = TTFF value
PIFFSMOD   = 2;       % model 1=Convex, 2=Linear, 3=Concave,
%                     4=convex-concave, 5=concave-convex
%**Machine Learning parameters
MLMODE     = 1;       % 0 = new, 1 = refresh
MLPITTFF   = 1;       % 0/1 = use PITTFF/learned
MLFDZ      = 1;       % 0/1 = as specified/learned
MLFFS      = 1;       % 0/1 = build using PITTFF/learned
% data-dependent values for plotting
AMA        = 10.0;    % Percent margin of accuracy
DATATT     = 'Demonstration';          %
FDXT       = 'Time [AU]';              % xaxis
FDYT       = 'FD [AU]';                % yaxis
RULYT      = 'RUL [AU]';               %
PHYT       = 'PH [AU]';                %
SOHYT      = 'SOH [%]';                %
FFPYT      = 'FFP Ratio [AU/AU]';      %
DPSYT      = 'DPS FFP XFRM [Ratio]';   %
FFSUYT     = 'FFS [%]';                %
FFSAYT     = 'Adjusted FFS [%]';       %
DUALYT     = 'RUL/PH [AU]';            % dual yaxis
```

Figure 6. Example of full list of input parameters for ARULE prior to simplification

```
%**Feature Data: FD = FDZ*(dP/P)^FDNV + DC + NOISE
FDC        = 24.0;    % Feature Data, DC
FDZ        = 0;       % Nominal FD0 value for AC coefficient: 0=use FDC
FDNM       = 5;       % Percent noise margin
FDCPTS     = 10;      % data points to average for FDC: up to 25
FDPTS      = 5;        % I: # data points to average for FD: up to 5
FDNV       = 1.275;    % Degradation power n
FFPFAIL    = 70.0;    % Failure margin - percent above nominal
%**Prognostic Modeling
PITTFF     = 220.0;   % Default RUL = TTFF value
PIFFSMOD   = 2;       % model 1=Convex, 2=Linear, 3=Concave,
%                      4=convex-concave, 5=concave-convex, 6=convex-concave
```

Figure 7. Example of full list of input parameters for ARULE after simplification

Figure 8 illustrates the concept of a predefined System Definition (SDEF) structure that points to two nodes, XFD1 and XFD2: the figure also illustrates the concept of a predefined Node Definition (NDEF) structure and Figure 10 illustrates the concept of API work areas in nonvolatile memory to update/save/restore definition and operational information.

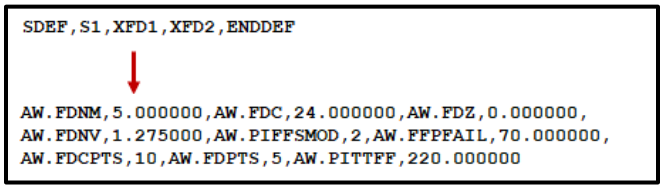


Figure 8. Example of pre-built structures

The microprocessor could be loaded with, for example, one to n predefined system definitions for specific solutions for prognostic applications. A select option could be provided for configurability. There are two primary marketing approaches: (1) APK microprocessors are pre-loaded as previously described, and/or (2) methods/means are made available for loading in and using user-defined definition structures: decision as to which (or even both) to be made later. The structures would be selected and loaded into non-volatile memory as part of an API area (see Figure 9 and Figure 10).

AW.PLSTAT,1,AW.FDNM,1.000000,AW.FDC,0.000000,AW.FDZ,1.000000,AW.FDNV,0.290000,AW.PIFFSMOD,6,AW.FFPFAIL,70.000000,AW.FDCPTS,0,AW.FDPTS,3,AW.PITTFF,30.000000,AW.cntfdc,0,AW.fdctot,0.000000,AW.fdc,0.000000,AW.cntfd,3,AW.totfd,2.625000e+00,AW.ixfd,1,AW.fd1,0.900000,AW.fd2,0.850000,AW.fd3,0.875000,AW.fd4,0.000000,AW.fd5,0.000000,AW.fdnom,1.000000,AW.fd,0.875000,AW.ffp,0.875000,AW.dps,0.630998,AW.fflv,0.292318,AW.ffs,215.860142,AW.ffin,215.860142,AW.ffq,0,AW.ffbd,0.000000,AW.ffap,0.000000,AW.fftp,0.000000,AW.rc0,0,AW.rs0,0,AW.rc1,0,AW.rs1,0,AW.t0,0.000000,AW.t1,2.940557,AW.t2,4.609400,AW.t3,3.838173,AW.a1,15.000000,AW.a2,55.000000,AW.a3,30.000000,AW.dt,24.000000,AW.da,215.860142,AW.rt,3.218877,AW.ra,25.159445,AW.dra,25.159445,AW.drt,3.218877,AW.st,3.838173,AW.sa,30.000000,AW.tst,11.388131,AW.tsa,0.000000,AW.dsp,3,AW.sdsp,3,AW.sdt,16.000000,AW.idpt,13,AW.ndsp,1,AW.pflag,3,AW.ptof,13.619296,AW.lsta,95.159445,AW.pdt,24.000000,AW.pda,0.000000,AW.dtoff,3.000000,AW.degrade,1,AW.modoff,3.000000,AW.bdtime,4.000000,AW.rul,0.000000,AW.soh,0.000000,AW.ph,0.000000,AW.bd,4.000000,AW.eol,17.000000,

Figure 9. Example API showing a node header

The example of an API work area shown in Figure 9 comprises the entire non-volatile memory required to process a single FD data point and produce estimates of prognostic information: data conditioning, data transforms, two stages of KF, and prognostic estimation. The area also comprises the necessary and sufficient information to support sampling intervals and checkpoint/restart.

B. Checkpoint/Restart

The software version of ARULE uses file storage to save and restore work areas between processing of sampled data. A method suitable for PHM SoC applications is to use non-volatile read/write memory to save and restore work areas. The memory could be segmented into blocks: one block for each node of the system (see Figure 10).

C. Data Acquisition and Output

Instead of accessing and outputting data from and into files, the APK microprocessor would have a front-end data interface to communicate with and exchange data with application-specific chips (Figure 10).

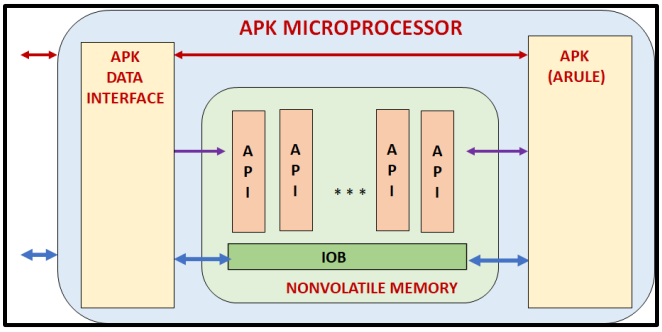


Figure 10. Concept: APK and nonvolatile memory

D. Eliminate Messaging and Logging

As is typical for software-based programs, the front-end program, UD_ARULE to the ARULE example of an APK issues messages: informational, warning, errors, and alerts to both a serial device and/or to a message log. For this effort, which does not comprise any textual visualization, messaging and logging is eliminated. The use of return and reason codes shall be retained in limited form: 0, 4, 8, and 12 return codes for, respectively, no error, warning, error, and severe and an accompanying reason, which shall be an identifier to where an error was detected. The return/reason codes are intended to aid in development.

4. Approach: Design, Develop, and Perform Testbenches

A. PHM SoC

We do not believe it is viable to design and develop a single PHM SoC if, for no other reason than the existence of sensors and sensor processing units (SPUs). The presented approaches allow for at least two approaches to use an APK microprocessor chip: (1) as a separately packaged product for use by customer-designed and developed applications and/or packaged as part of a multiple chip module as a plug-in application-specific solution: see Figure 11.

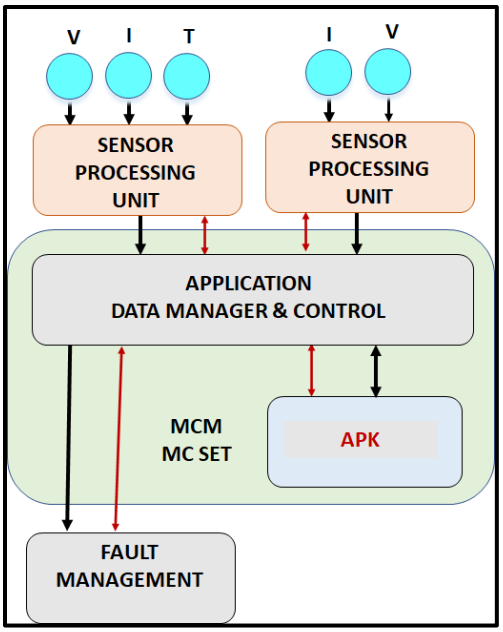


Figure 11. Concept: PHM SoC comprising multiple sensors to be controlled and managed as a single system

B. Test Benches

A test bench (TB) is used to verify and validate correct operation for both software and firmware. Using Figure 11 as a guide, the following is an approach for bench testing and for design and development of a chip set for demonstration of proof:

- Design, develop, and perform a TB for the APK Chip. The TB needs to comprise the functionality of an application-specific Data Manager and Control (DMC) as shown in Figure 12.

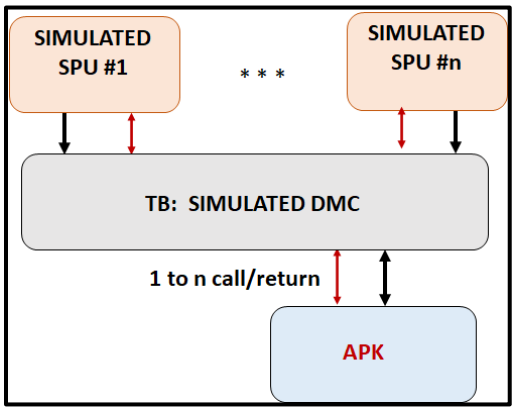


Figure 12. TB diagram for an APK-on-Chip

- Convert the TB for the APK chip into an application-specific DMC that interfaces to the APK chip.
- Design, develop, and perform a TB for a DMC-on-chip connected to the APK-on-chip as illustrated in Figure 13.

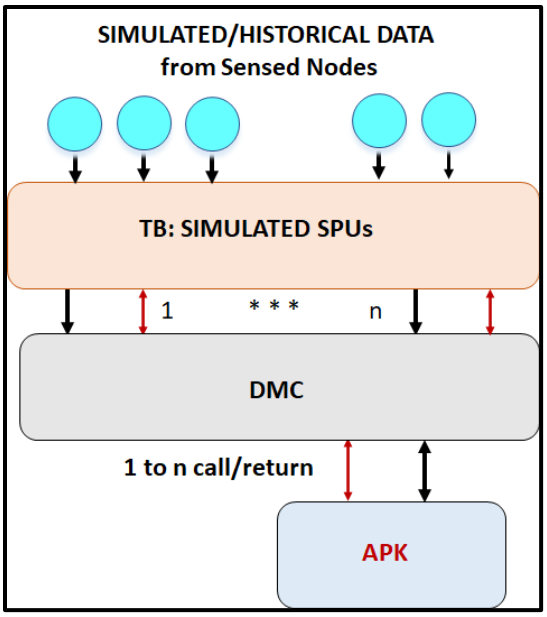


Figure 13. TB diagram for connected DMC and APK

- Convert the TB for the APK chip into an application-specific DMC that interfaces to the APK chip.
- Design, develop, and perform a TB for the SPUs as illustrated in Figure 14.

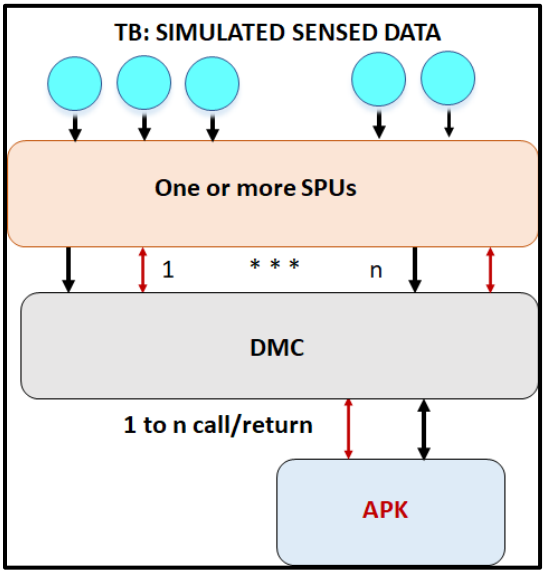


Figure 14. TB diagram for one or more SPUs connected to one or more DMCs connected to an APK

5. Approach: Design, Develop, and Perform Demonstration Proof of PHM SoC

We choose a hand-launched, commercial quadcopter to be prognostic enabled for a demonstration proof of the PHM SoC in general and the APK-on-chip specifically: see Figure 15.

A. *Demonstration: Prognostic-enabling Approaches*

Prognostic-enabling approaches include the following: (1) sensor at each rotating assembly (RA) to detect excessive and/or unbalanced vibration forces indicative of problems with the RA motor, strut-mounting, blade, and/or shaft; and (2) a centric-mounted sensor to detect problems with one or more RAs.

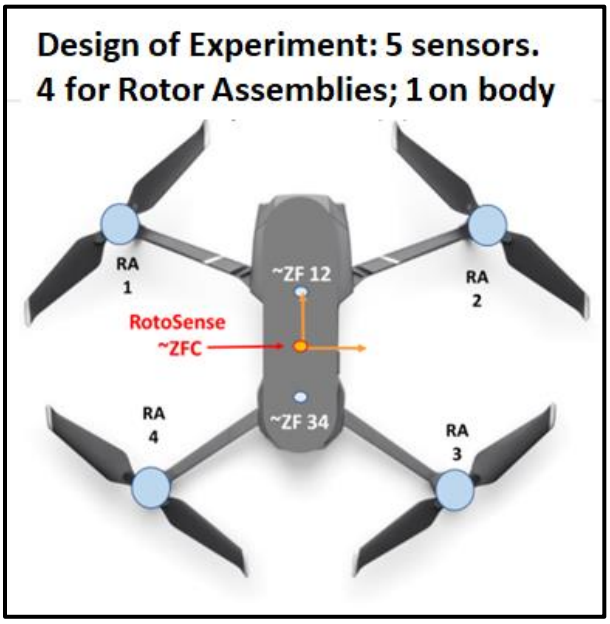


Figure 15. Demonstration concept: design of experiment for a quadcopter

B. *RA Sensor Approaches*

The RA sensor approach uses a microelectromechanical system (MEMS): to detect and collect vibration-force data. When the quadcopter is on the verge of lifting or at hover, the forces for each RA should be (1) at or below a minimum lift-off threshold and (2) the force for each RA should be approximately equal to its nearest neighbors.

The centric-mounted sensor is believed to be necessary to detect when the quadcopter is just below the lift force for vertical flight: no forward-backward and no sideways movement. This is the preflight zone during which the RA sensors should be sensed.

C. *Design of Experiment*

The experiment design is to mount a MEMS-based, triaxial-sensor unit to each RA to monitor and collect vibration data and then wirelessly transmit that data to a ground-based hub. Another MEMS-based, triaxial-sensor or inertial measurement unit (IMU) shall be mounted on the top of the body of the quad-copter to also monitor, collect, and transmit vibration data.

A to-be-determined number of cyclic regimes shall be run:

- All RAs undamaged.
- Data analyses to characterize data for undamaged states: at rest idling, increasing blade speed from idle to hovering.
- Damaged RA for at least one of the following: blade damage, shaft binding, bent strut, and degraded motor.
- Data analyses to characterize data for varying levels of damage and to determine the efficacy of a centric-mounted sensor either in lieu of or in addition to RA sensors.
- Design a fault-detection display method for a demonstration to prove successful APK-on-chip in a PHM SoC
- Run a two-stage demonstration: (1) undamaged and (2) damaged.

6. SUMMARY AND CONCLUSION

This paper presented an introduction to an example APK (Figure 1) primarily comprising the SD, HA, and PA elements of IEEE 1856-2017 with some front-end DM (Figure 2). The APK was then shown embedded in an example PHM system (Figure 3) and configured in a software-based TB (Figure 4 and Figure 5).

Approaches to realize successful export of the example APK to an APK-on-chip for use in a PHM SoC included the following: (1) simplify the example APK by eliminating unneeded/unwanted options, complex configuration and customization, and file-based messages and logging; (2) design, develop, and perform TB to accomplish a step-wise design and development; and (3) a final approach comprising the design, development, and performance of a demonstration to prove APK-on-chip operation in a PHM SoC. For the demonstration, we plan on prognostic-enabling a hand-launched quadcopter: each of the RAs and a centric-mounted, MEMS-based sensor on the top of the body of the quadcopter.

We believe the approaches presented in this paper shall result in successful export of a software-based APK to an APK-on-chip solution in a PHM SoC, UD_ARULE, and the ARULE kernel to be revised and loaded into a microprocessor chip to support PHM SoC.

ACKNOWLEDGEMENTS

The authors thank Naval Air, Naval Sea, U.S. Army, U.S. Air Force, NASA research centers, and Idaho National Laboratories for their support and funding of multiple projects that led to the design, development, and commercialization of ARULE: the ARULE GUI, UD_ARULE, and the ARULE kernel to be revised and loaded into a microprocessor chip to support PHM SoC.

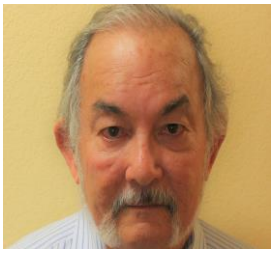
REFERENCES

[1] 1856-2017 IEEE Standard Framework for Prognosis and Health Management of Electronic Systems, Dec. 2017.

[2] Goodman, Douglas L.; Hofmeister, James P.; and Szidarovszky, Ferenc, 2019; Prognostics and Health Management: A Practical Approach Using Conditioned-Based Data, 1st Edition, John Wiley & Sons, Ltd, The Atrium, Southern Gate, Chichester, West Sussex, PO19 8SQ, United Kingdom, ISBN: 9781119356653, 2019.

[3] Hofmeister, J., Pena, W., and Curti, C., 2023; "Reducing Signature Models for Extended Kalman Filtering for Adaptive Prognostic Estimation," 2023 IEEE Aerospace Conference, Big Sky, MT, 4-11 March, 2023.

BIOGRAPHY



James Hofmeister *received an M.S. in Elec. & Computer Engineering from the University of Arizona, Tucson, (2002), a B.S. in Electrical Engineering from the University of Hawai'i, Honolulu (1993), and is a graduate of IBM's Systems Research Institute (1986): Science and Engineering of Computer-oriented Systems. After a 30-year career with IBM and five years of retirement, he joined Ridgetop Group, Inc. in 2003. He is currently a Distinguished Engineer and Principal Investigator in diagnostic and prognostic health monitoring/management systems. His accomplishments include seven issued U.S. patents; and over 50 published papers, articles, and a book in the Wiley Reliability Engineering series. He is an IEEE Senior Life Member, and an elected Fellow of the Society for machinery Failure Prevention Technology (MFPT), an emeritus member of the Board of Directors of MFPT, and is a member of the IEEE Reliability Society*

Wyatt Pena *received a B.S. in Engineering Management with a technical minor in Systems Engineering from the University of Arizona (2017). Wyatt is the Director of Operations at Ridgetop Group and has been with the company for approximately 6+ years. By utilizing his strong background in systems engineering and project management, Wyatt oversees day-to-day operations to ensure that engineering and business development activities are in direct alignment with the interest of Ridgetop's customers, shareholders, and overall company mission. Wyatt has played a key role in managing all aspects of product design, development, and deployment of multifaceted Ridgetop solutions comprising hardware, firmware, and software elements. Wyatt has most recently been leading the transition of Ridgetop IP to commercial products and solutions that are being utilized around the globe. Prior to becoming the Director of Operations, Wyatt has served as a Project Manager, Systems Engineer, and Test Engineer on numerous government and commercial contracts. Other core strengths include an in-depth understanding in engineering management, cost estimation, supply chain management, as well as prototype design and manufacturing of CBM, PHM, and IVHM solutions.*

Christopher Curti *has two BS degrees from the University of Arizona: Molecular & Cellular Biology and Electrical & Computer Engineering. He joined Ridgetop Group, Inc in 2019 and is an experienced software engineer and contributes to the design and development of commercial PHM and CBM software products and solutions.*