FLIGHT SOFTWARE DEVELOPMENT WITH TASTE

Steve Duncan

MBSE 2022

Date: 23/11/2022
Ref: TASUK/19280/PRS/001
Template: 83230347-DOC-TAS-EN-006

INTRODUCTION

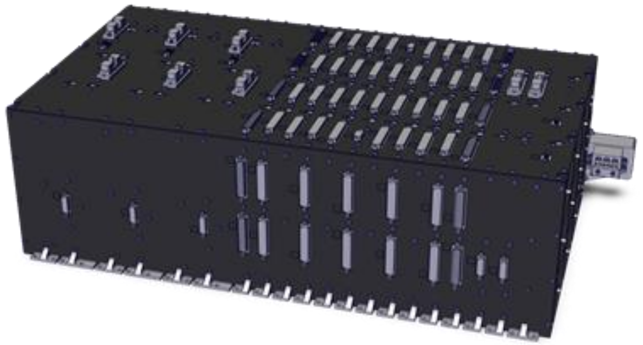
- Space Inspire is a mid-sized communications satellite solution that complements the existing SpaceBus NEO product line
- Extremely high capacity, agility, in-orbit reconfiguration
- Designed for very competitive price point
- Platform for ASTRA 1Q, SES-26, ARABSAT 7A, Intelsat 41 & 44

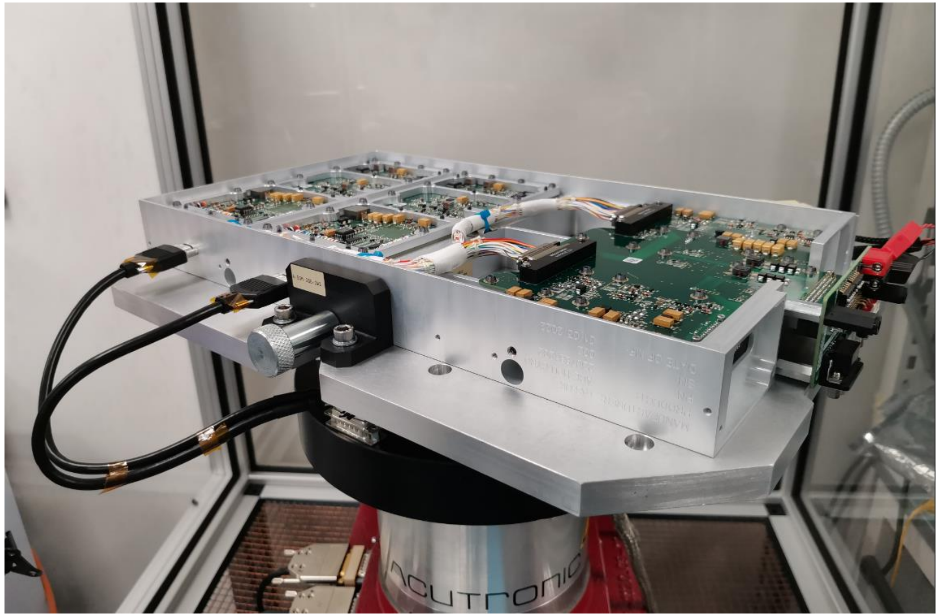
(c) Thales Group

Date: 23/11/2022
Ref: TASUK/19280/PRS/001
Template: 83230347-DOC-TAS-EN-006

SPACE INSPIRE GYRO

- Part of the ACE-HPU modular avionics subsystem
  - **Multipurpose I/O**
  - **Secondary power distribution**
  - **Auxiliary propulsion control**
  - **High power conditioning and distribution to PF/PL**
  - **High voltage control and distribution for electric propulsion**
- Cost reduction through use of upscreened industrial and automotive components
- Use of rad-hard ICs in low-cost packages



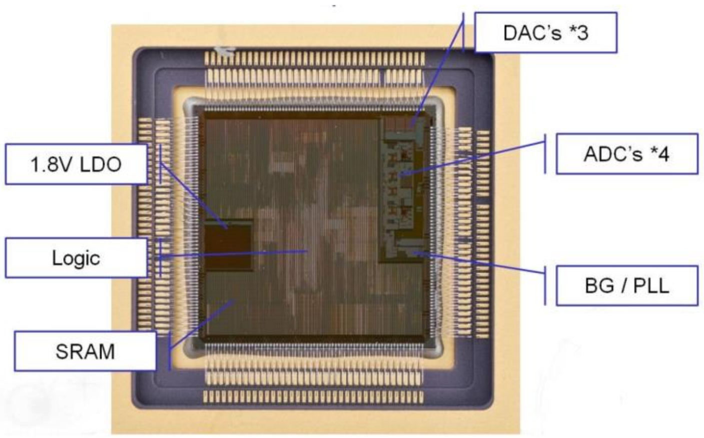


Date: 23/11/2022
Ref: TASUK/19280/PRS/001
Template: 83230347-DOC-TAS-EN-006

DPC MICROCONTROLLER

○ Digital Programmable Controller (DPC)

- **Radiation Hardened**
- **Low FM cost (BGA package)**
- **High Performance**
- **Large array of on-chip peripherals**
- **Full tool chain and driver support**

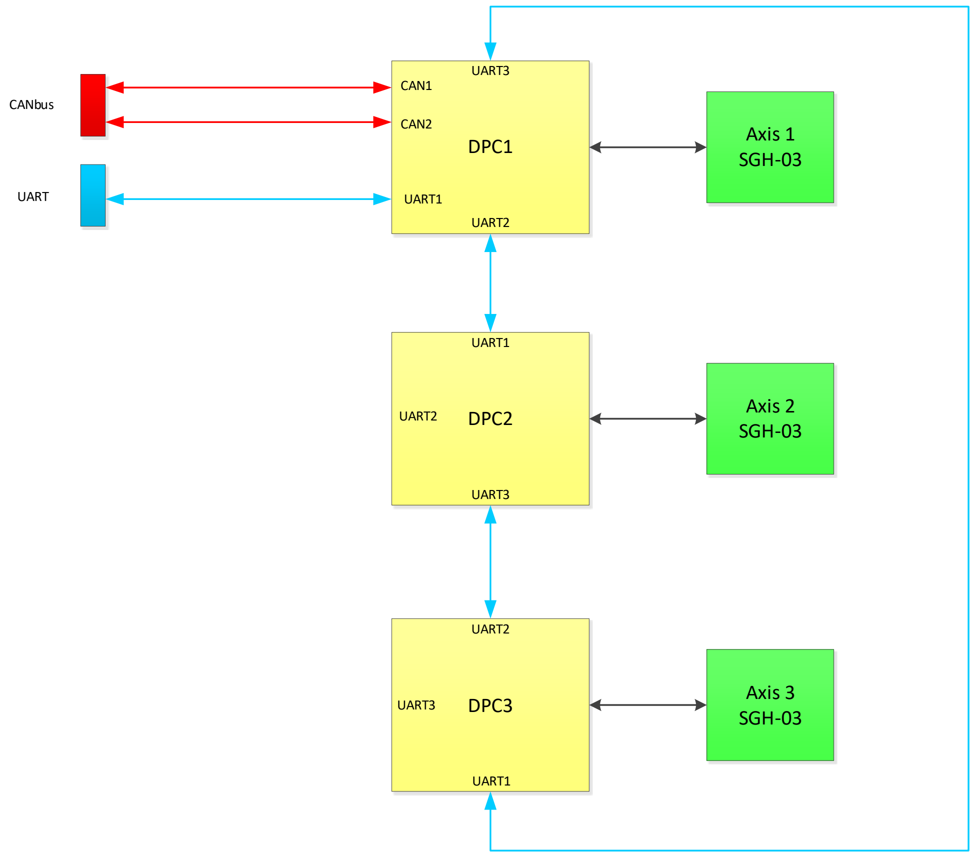


Manufacturer	Thales Alenia Space
CPU	MSP430
Word size	16
Clock	15-60 MHz
Cores	3
Endianism	little
Program/Data RAM (kB)	28 / 14
PROM	External SPI NVRAM
ADC	4 x 13 bit
DAC	3 x 12 bit
Mil-1553B	✓
CANbus	✓
SpaceWire	×
Radiation class	Rad-hard
FM availability	✓

Date: 23/11/2022
Ref: TASUK/19280/PRS/001
Template: 83230347-DOC-TAS-EN-006

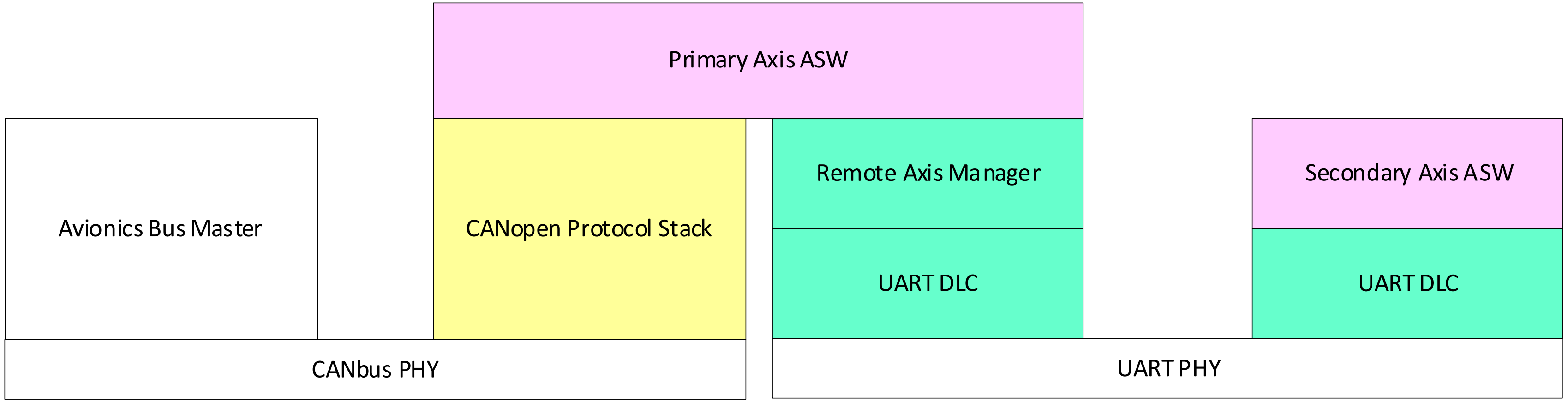
GYRO ARCHITECTURE

- Single SGH-03 device per axis
- Each axis has a dedicated DPC microcontroller
  - **RAS, SSM cores perform detector control**
  - **COM core handles communications**
- External bus connection is CANbus or (optionally) UART
- Internal inter-axis communications using UART



Date: 23/11/2022
Ref: TASUK/19280/PRS/001
Template: 83230347-DOC-TAS-EN-006

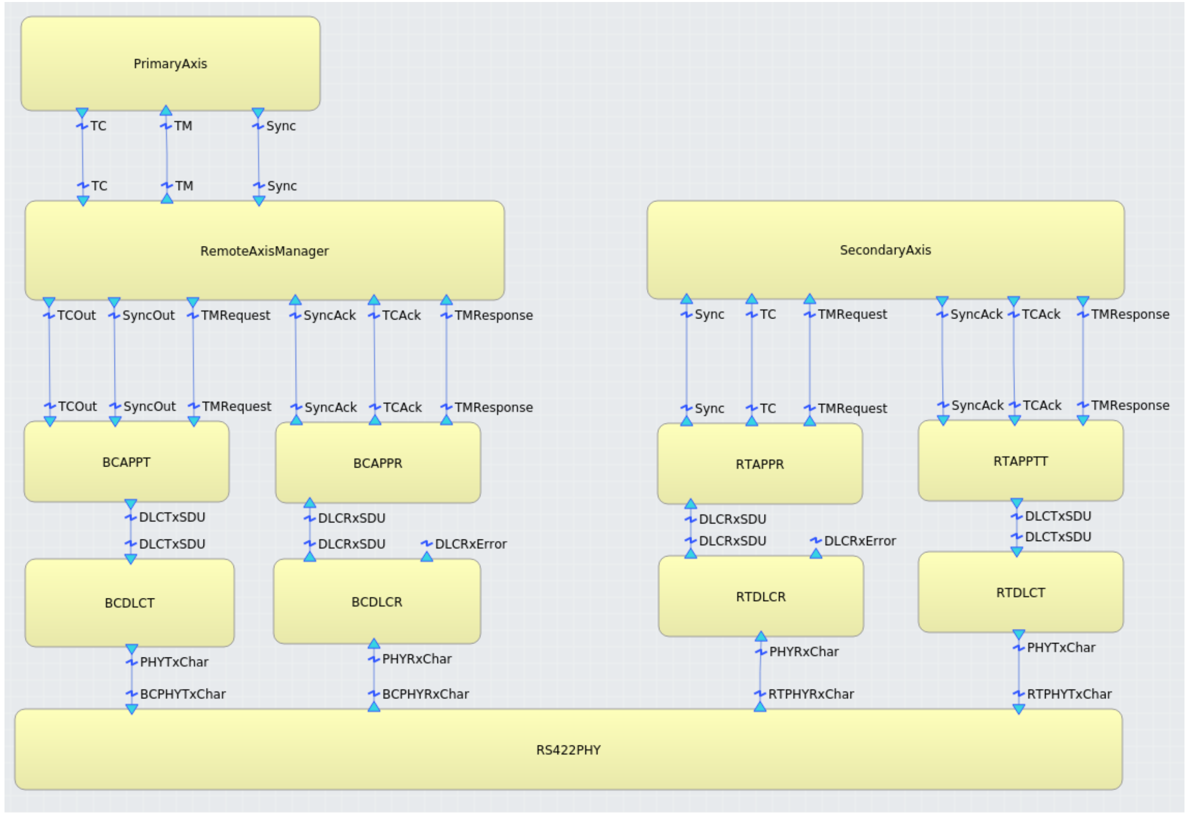
SW LAYERS (COM)



🟪	Manually coded
🟨	Autogenerated, replaceable
🟩	Autogenerated

Date: 23/11/2022
Ref: TASUK/19280/PRS/001
Template: 83230347-DOC-TAS-EN-006

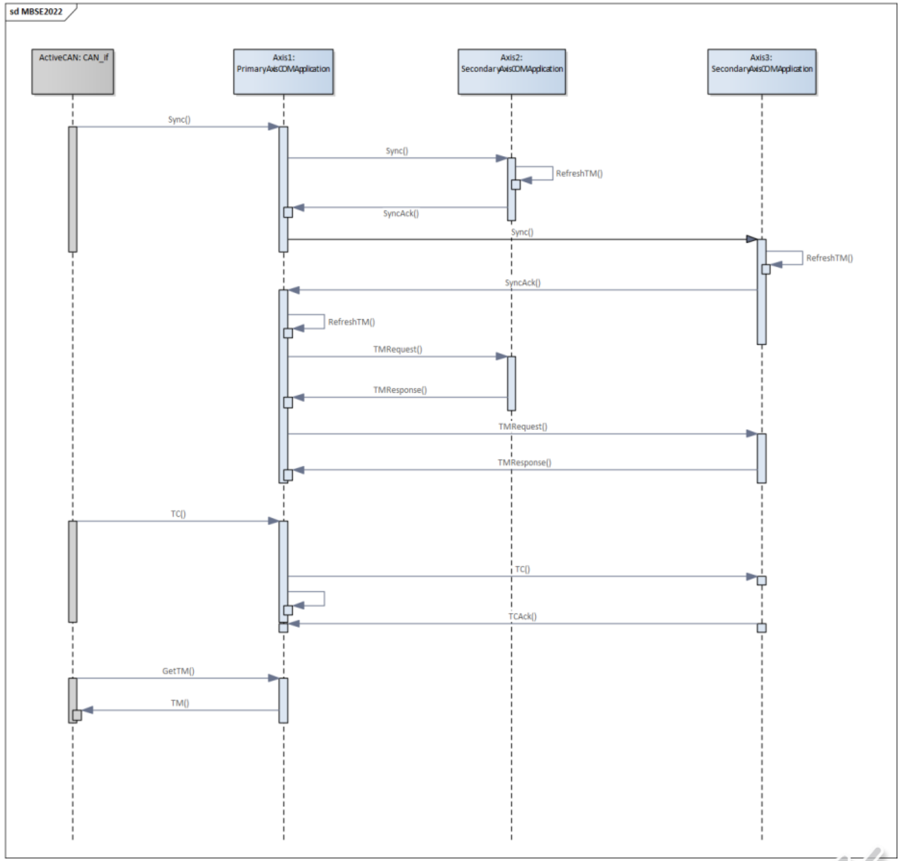
TASTE AADL MODEL



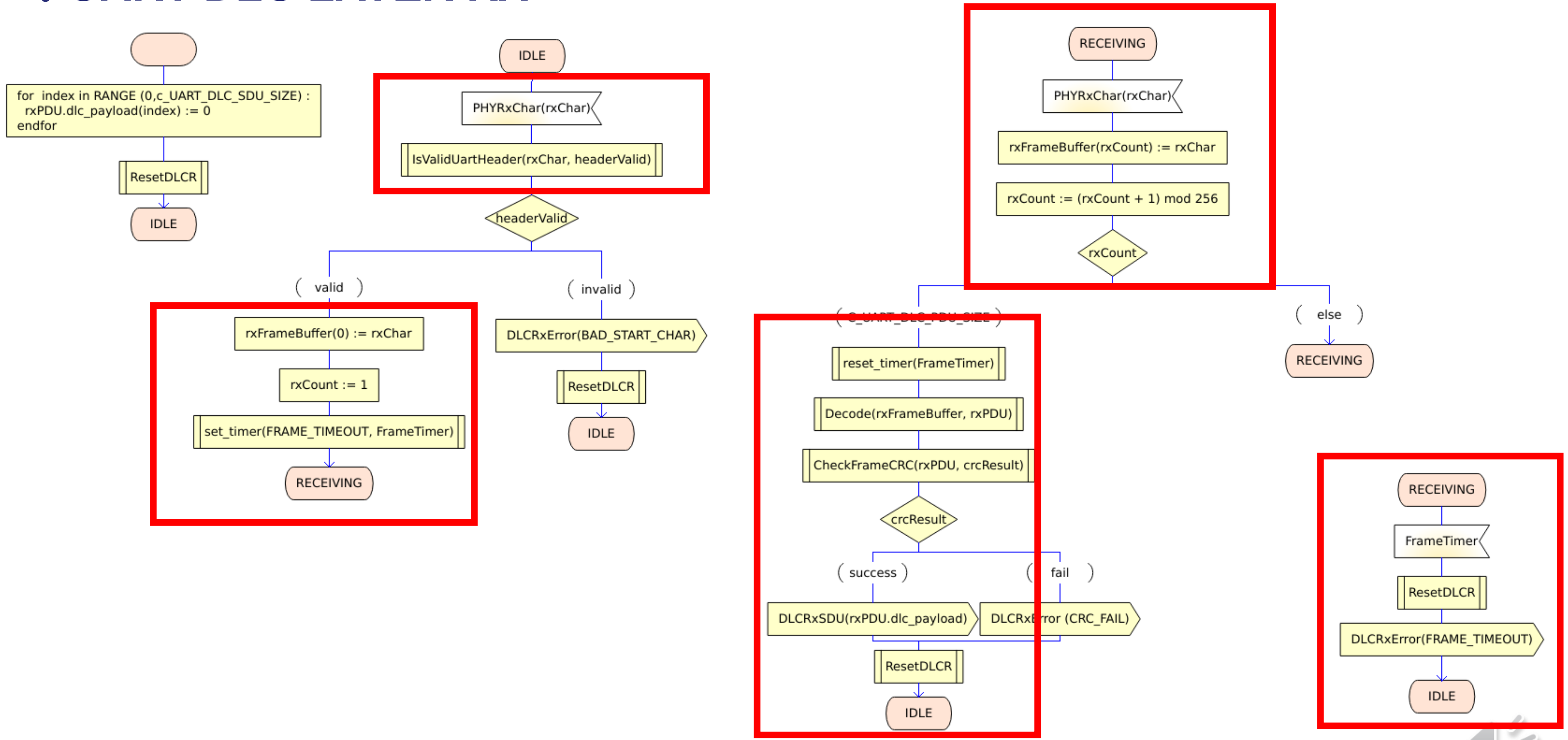
Date: 23/11/2022
Ref: TASUK/19280/PRS/001
Template: 83230347-DOC-TAS-EN-006

DYNAMIC ARCHITECTURE

- Inter-axis communication adds significant complexity through the addition of new failure paths
- Robustness must be designed in from the start
  - **rejection of messages with bit errors**
  - **Detection of lost characters**
  - **recovery message loss**
- Formal specification of the protocols in SDL
  - **SDL can be analysed, executed and proven**

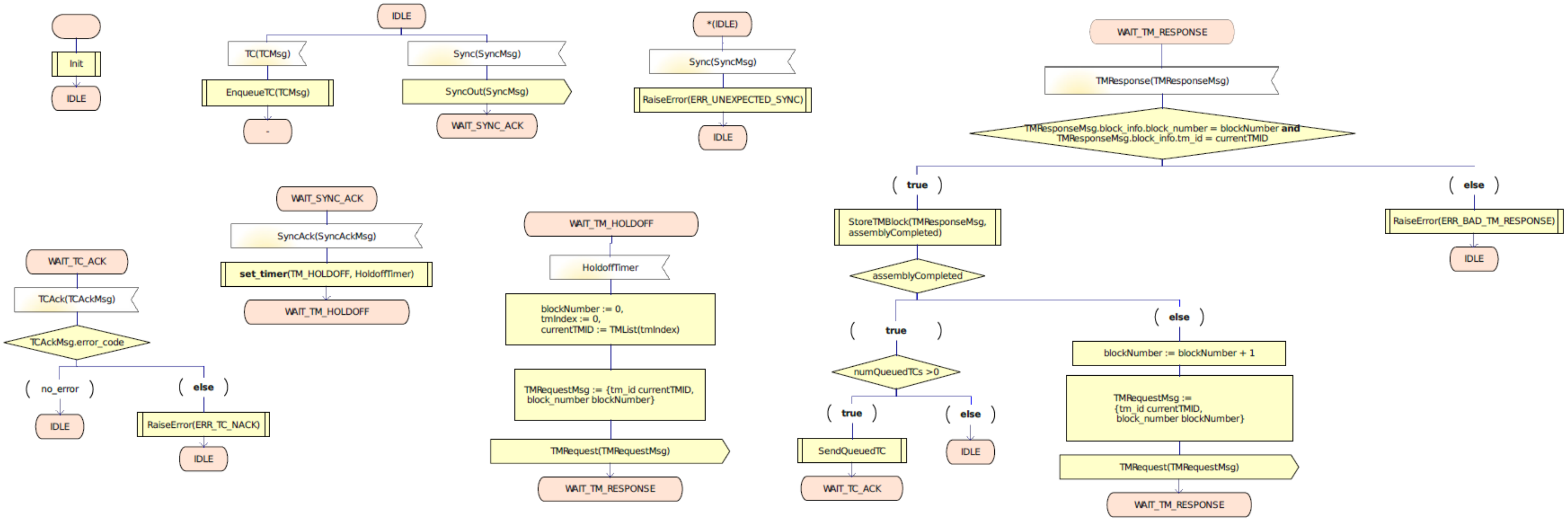


UART DLC LAYER RX



for index in RANGE (0,c_UART_DLC_SDU_SIZE) :
rxPDU.dlc_payload(index) := 0
endfor
ResetDLCR
IDLE
IDLE
PHYRxChar(rxChar)
IsValidUartHeader(rxChar, headerValid)
headerValid
valid
rxFrameBuffer(0) := rxChar
rxCount := 1
set_timer(FRAME_TIMEOUT, FrameTimer)
RECEIVING
invalid
DLCRxError(BAD_START_CHAR)
ResetDLCR
IDLE
RECEIVING
PHYRxChar(rxChar)
rxFrameBuffer(rxCount) := rxChar
rxCount := (rxCount + 1) mod 256
rxCount
C_UART_DLC_PDU_SIZE
reset_timer(FrameTimer)
Decode(rxFrameBuffer, rxPDU)
CheckFrameCRC(rxPDU, crcResult)
crcResult
success
DLCRxSDU(rxPDU.dlc_payload)
fail
DLCRxError (CRC_FAIL)
ResetDLCR
IDLE
else
RECEIVING
RECEIVING
FrameTimer
ResetDLCR
DLCRxError(FRAME_TIMEOUT)
IDLE

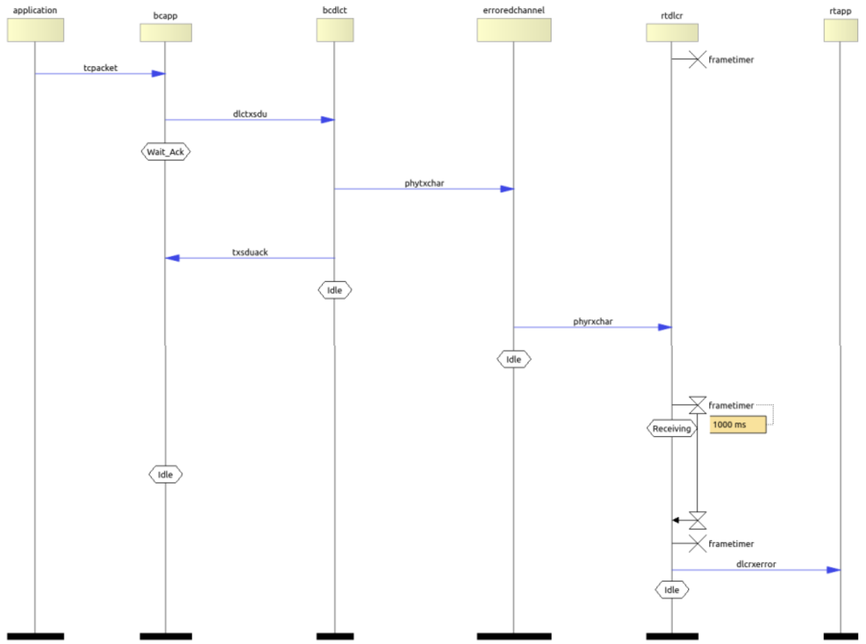
REMOTE AXIS MANAGER


Init
IDLE
IDLE
TC(TCMsg)
EnqueueTC(TCMsg)
-
Sync(SyncMsg)
SyncOut(SyncMsg)
WAIT_SYNC_ACK
*(IDLE)
Sync(SyncMsg)
RaiseError(ERR_UNEXPECTED_SYNC)
IDLE
WAIT_TM_RESPONSE
TMResponse(TMResponseMsg)
TMResponseMsg.block_info.block_number = blockNumber and TMResponseMsg.block_info.tm_id = currentTMID
(true)
StoreTMBlock(TMResponseMsg, assemblyCompleted)
assemblyCompleted
(true)
numQueuedTCs > 0
(true)
SendQueuedTC
WAIT_TC_ACK
(else)
IDLE
(else)
blockNumber := blockNumber + 1
TMRequestMsg := {tm_id currentTMID, block_number blockNumber}
TMRequest(TMRequestMsg)
WAIT_TM_RESPONSE
(else)
RaiseError(ERR_BAD_TM_RESPONSE)
IDLE
WAIT_SYNC_ACK
SyncAck(SyncAckMsg)
set_timer(TM_HOLDOFF, HoldoffTimer)
WAIT_TM_HOLDOFF
WAIT_TC_ACK
TCAck(TCAckMsg)
TCAckMsg.error_code
(no_error)
IDLE
(else)
RaiseError(ERR_TC_NACK)
IDLE
WAIT_TM_HOLDOFF
HoldoffTimer
blockNumber := 0, tmIndex := 0, currentTMID := TMList(tmIndex)
TMRequestMsg := {tm_id currentTMID, block_number blockNumber}
TMRequest(TMRequestMsg)
WAIT_TM_RESPONSE

Date: 23/11/2022
Ref: TASUK/19280/PRS/001
Template: 83230347-DOC-TAS-EN-006

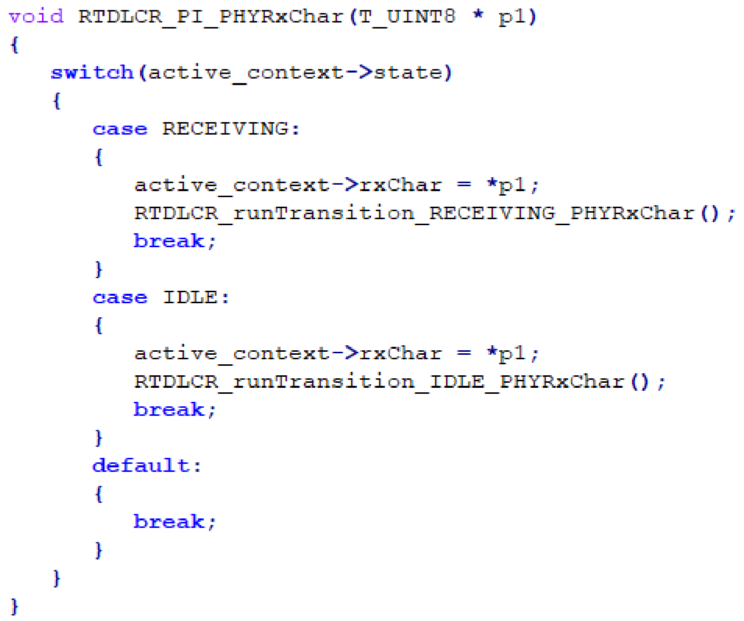
DESIGN VERIFICATION

- SDL is first verified by inspection
- Test cases are then identified that will invoke all FSM transitions
- Test cases are executed on the end-to-end model within the TASTE VM
  - **Manual control possible via TASTE GUI**
  - **Sessions saved as Python scripts and replayed**
- Animation of the system behaviour is key benefit of TASTE
  - **Design is verified before any implementation commences**



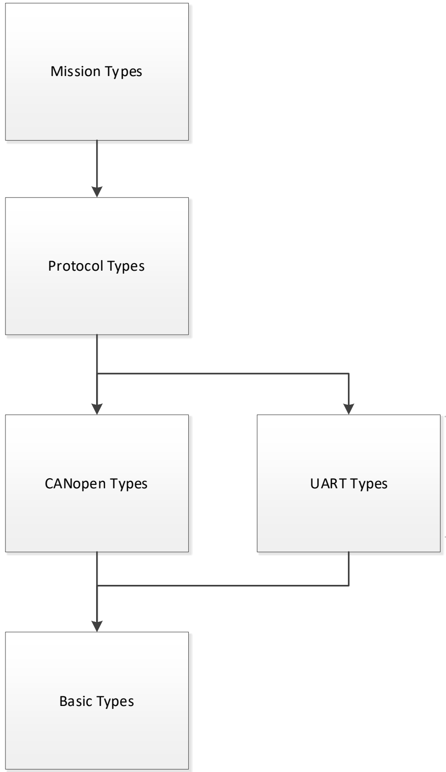
AUTOGENERATED FSM CODE

- Computational Model
  - **Transitions driven by Provided Interface (PI) calls**
  - **Transitions are atomic, code is not re-entrant**
  - **Timers do not interrupt transitions**
  - **Outputs conducted via Required Interface (RI) calls**
  - **RIs may be executed in context of calling process, or asynchronously by IPC (user's choice)**
- OpenGeode code generator mods
  - **Typographic case used in SDL identifiers is preserved**
  - **Direct access to context structure is replaced by pointer mechanism**
  - **runTransition() function is simplified, "next transition" removed**
  - **Functions specific to the TASTE VM are omitted, e.g. get_sender()**
- Autogenerated code is subject to the same verification as manually generated code



```
void RTDLCR_PI_PHYRxChar(T_UINT8 * p1)
{
    switch(active_context->state)
    {
        case RECEIVING:
        {
            active_context->rxChar = *p1;
            RTDLCR_runTransition_RECEIVING_PHYRxChar();
            break;
        }
        case IDLE:
        {
            active_context->rxChar = *p1;
            RTDLCR_runTransition_IDLE_PHYRxChar();
            break;
        }
        default:
        {
            break;
        }
    }
}
```

ASN.1 DATA MODEL



```
------------------------------------------------------------------------------------
-- UART DLC Layer Definitions
------------------------------------------------------------------------------------

c-UART-DLC-SYNC-VALUE UINT8 ::= 85

T-UART-DLC-Header   ::= UINT8(c-UART-DLC-SYNC-VALUE)
T-UART-DLC-FrameCRC ::= UINT8

T-UART-DLC-SDU ::= SEQUENCE (SIZE(c-UART-DLC-SDU-SIZE)) OF UINT8

T-UART-DLC-PDU ::= SEQUENCE {
dlc-header   T-UART-DLC-Header,
dlc-payload  T-UART-DLC-SDU,
crc          T-UART-DLC-FrameCRC
}

T-UART-DLC-RxErrorCode ::= ENUMERATED {
    no-error (0),
    bad-start-char (1),
    crc-fail (2),
    frame-timeout (3),
    bad-request-type (4),
    phy-rx-error(5)
}
```

ASN.1 CODE GENERATION

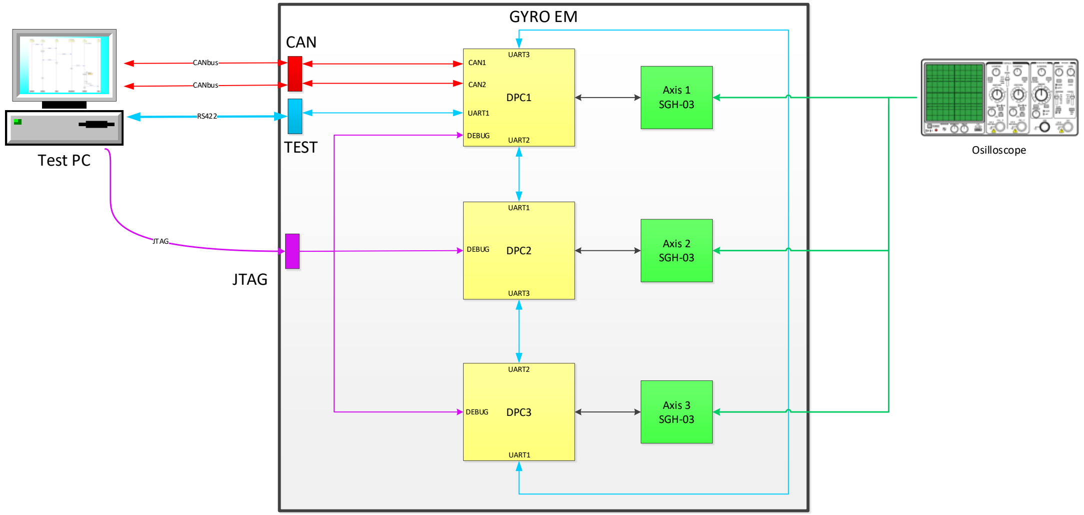
- ASN1SCC compiler output is too large for µCs
  - **Approx. 14 KLOC for minimal CANopen implementation**
  - **240kB object code (LEON)**
- Alternative encoding approach needed
  - **Use ASN1Scc XML Abstract Syntax Tree to generate C structures with implicit line encoding**
  - **Minimal protocol compiles to < 3kB**
  - **Independent of processor bus width or endianism**
- Some constraints apply
  - **Encoded structures may not contain optional components**

```
T-CANopen-SDO-Rx ::= CHOICE {
  downloadInitiate            T-CANopen-SDO-DownloadInitiate,
  uploadInitiate              T-CANopen-SDO-UploadInitiate,
  downloadSegment             T-CANopen-SDO-DownloadSegment,
  uploadSegment               T-CANopen-SDO-UploadSegment,
  downloadAbort               T-CANopen-SDO-Abort
}

T-CANopen-SDO-Tx ::= CHOICE {
  uploadInitiateResponse      T-CANopen-SDO-UploadInitiateResponse,
  downloadInitiateResponse    T-CANopen-SDO-DownloadInitiateResponse,
  downloadSegmentResponse     T-CANopen-SDO-DownloadSegmentResponse,
  uploadSegmentResponse       T-CANopen-SDO-UploadSegmentResponse,
  uploadAbort                 T-CANopen-SDO-Abort
}

T-CANopen-NMT-ErrorControl ::= CHOICE {
  heartbeat                   T-CANopen-Heartbeat
}

------------------------------------------------------------------------
--  CANopen frame definition
------------------------------------------------------------------------

T-CANopen-Header ::= SEQUENCE {
  functionCode T-CANopen-FunctionCode,
  nodeID       T-CANopen-NodeID,
  rtr          T-CANopen-RTR,
  dlc          T-CANopen-DLC
}

T-CANopen-Payload ::=  CHOICE {
  emergency         T-CANopen-Emergency,
  sdo-tx            T-CANopen-SDO-Tx,
  sdo-rx            T-CANopen-SDO-Rx,
  nmt-error-control T-CANopen-NMT-ErrorControl
}

T-CANopen-Frame ::= SEQUENCE {
  header       T-CANopen-Header,
  payload      T-CANopen-Payload
}
```

```
/* ASN.1 typename: T-CANopen-SDO-Rx */
/* ASN.1 Type T-CANopen-SDO-Rx, packed size: 64 bits */
typedef union {
  T_CANopen_SDO_DownloadInitiate downloadInitiate;
  T_CANopen_SDO_UploadInitiate   uploadInitiate;
  T_CANopen_SDO_DownloadSegment  downloadSegment;
  T_CANopen_SDO_UploadSegment    uploadSegment;
  T_CANopen_SDO_Abort            downloadAbort;
} T_CANopen_SDO_Rx;

/* ASN.1 typename: T-CANopen-SDO-Tx */
/* ASN.1 Type T-CANopen-SDO-Tx, packed size: 64 bits */
typedef union {
  T_CANopen_SDO_UploadInitiateResponse   uploadInitiateResponse;
  T_CANopen_SDO_DownloadInitiateResponse downloadInitiateResponse;
  T_CANopen_SDO_DownloadSegmentResponse  downloadSegmentResponse;
  T_CANopen_SDO_UploadSegmentResponse    uploadSegmentResponse;
  T_CANopen_SDO_Abort                    uploadAbort;
} T_CANopen_SDO_Tx;

/* ASN.1 typename: T-CANopen-NMT-ErrorControl */
/* ASN.1 Type T-CANopen-NMT-ErrorControl, packed size: 8 bits */
typedef union {
  T_CANopen_Heartbeat heartbeat;
} T_CANopen_NMT_ErrorControl;

/* ASN.1 typename: T-CANopen-Header */
/* Packed size: 16 bits */
typedef struct {
  T_CANopen_FunctionCode functionCode:4;
  T_CANopen_NodeID       nodeID:7;
  T_CANopen_RTR          rtr:1;
  T_CANopen_DLC          dlc:4;
} __attribute__((packed)) T_CANopen_Header;

/* ASN.1 typename: T-CANopen-Payload */
/* ASN.1 Type T-CANopen-Payload, packed size: 64 bits */
typedef union {
  T_CANopen_Emergency        emergency;
  T_CANopen_SDO_Tx           sdo-tx;
  T_CANopen_SDO_Rx           sdo-rx;
  T_CANopen_NMT_ErrorControl nmt-error-control;
} T_CANopen_Payload;

/* ASN.1 typename: T-CANopen-Frame */
/* Packed size: 80 bits */
typedef struct {
  T_CANopen_Header  header;
  T_CANopen_Payload payload;
} __attribute__((packed)) T_CANopen_Frame;
```

SW INTEGRATION / VALIDATION

- CANopen bus master SDL model executed within TASTE VM
  - **Sub real-time execution**
  - **Allows interactive debugging with inspection and modification of message sequence**
- TASTE component interfaces to physical CANbus via USB
- JTAG also used for inspection of DPC state and for provocation of errors
  - **UART bit errors**
  - **Bus failure**
  - **ADC/DAC SEU/SEFI**
- JTAG functionality integrated with TASTE scripting engine



CONCLUSIONS

- MBSE gives savings across the life cycle, from design through to qualification
- Early design verification is a key contributor to reliability and also instils confidence in stakeholders
- Use of SDL for behavioural specification is far superior to other means
- Autogeneration increases portability by ensuring models are expressed in machine and language independent format
- Models are exchangeable and reusable and indeed are already baselined for future equipment projects

THANK YOU

stephen.duncan@thalesaleniaspace.com